趣味の技術入門

●よくわかる

マイコンの使い方遊び方

谷本敏一
野島久雄

新星出版社

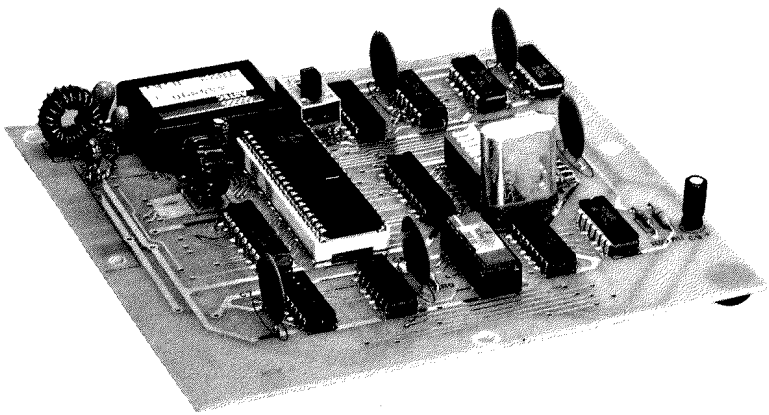
● **よくわかる マイコン入門**――山科敦之・吉田紀彦著
マイコンの仕組み、基本的な働き、取扱う情報などをわかりやすくまとめた。

● **よくわかる マイコンの使い方遊び方**―谷本敏一・野島久雄著
マイコンの基礎的知識、ライフゲームなどのプログラムと遊び方を紹介した。

● **初歩のマイコン入門**――佐藤清明・三宅 誠著
マイコンを自由に使うための考え方、技術を、プログラムを中心に解説した。

BASIC FOR MICROCOMPUTERS

よくわかる マイコンの使い方遊び方



はじめに

コンピュータがこの世に誕生したのは、今からほぼ40年前のことです。その頃のコンピュータは、現在のコンピュータに比べて、能力が限られたものでした。そのうえ、製作に莫大な費用がかかり、取扱いもやっかいな巨大な電子装置でした。それにもかかわらず、コンピュータを作った人々は、コンピュータがやがて世界を変えるだろう、という希望と自信を持っていたようです。コンピュータが「電子頭脳」、「人工知能」と呼ばれる時代がくるという夢があったのかもしれません。

その後、さまざまな科学技術の進歩によって、コンピュータにかけられた期待のいくつかは実現されています。現代社会は、コンピュータなしでは成立たないと言われるほど、コンピュータの利用は進んでいるのです。

本書で私たちが学ぼうとしているマイコンは、まだ生まれてから数年ですが、コンピュータの発達の歴史の中で、ひとつの革命を起こしました。それは、コンピュータを「私たち」のものにできるということです。すばらしい能力を持った小さなコンピュータを手頃な価格で手に入れ、自分専用のコンピュータとして使えるとい

うことは、コンピュータが生まれた頃、人々が描いた夢のひとつです。

「誰にでも使える」コンピュータは、使い方がやさしくなければなりません。そのために、ベーシックという言葉が開発されました。本書では、このベーシックという言葉を用いてマイコンを使いこなす方法を解説していきます。

筆者たちは、コンピュータを使う立場の人間です。コンピュータの使い方（ソフトウェア）を研究してきた「ソフト屋」です。ですから、マイコンの仕組（ハードウェア）よりも、実際の活用例を中心に、本書を組立てました。

マイコンは、誰にでも使えるはずです。本書によって、みなさんが、未来のコンピュータ社会を積極的に築いていく仲間に加わってくださったら、嬉しく思います。

終りに、本書の上梓に協力してくださった森耕一氏、東京大学文学部吉野諒三君、津崎実君、読者の立場からアドバイスしていただいた日大豊山中学江島君そして、快く写真を提供してくださったメーカー、販売店各社に感謝します。

谷本敏一
野島久雄

目次

第1章　マイコンは、だれにでも使える……9

マイコンの構造は、わからないもの……10

マイコンはシステム……13

システムの実例（ワンボード・マイコン　マイコン・セット）

マイコンの周辺機器……18

CRTディスプレイ（画面の表示　文字の表示　ドット方式）／キーボード（JISとASCII　キーボードの使い方　シフト・キーの使い方　文章の入力　「復改」キー　「後退」キー　「まっ消」キー　マイコンの答）／プリンタ／カセット・テープ／フロッピーディスク／そのほかの周辺機器（タイプライタ　紙テープ・リーダ、パンチャー　XYプロッタ）

第2章　ベーシックによるプログラミング……37

マイコンとプログラム……38

プログラムとは／プログラムの流れ／フローチャートの書き方

ベーシックとは……47
なぜベーシックを使うか／ベーシックの使い方／ベーシックのプログラム／ベーシックの文（行番号　PRINT　INPUT　LET　変数　IF—THEN　GOTO　ON—GOTO　GOSUB　FOR—NEXT　READ・DATA・RESTORE　REM　DIM　組込み関数）
マイコンの働き……69
ベーシック・インタプリタ（プログラムと制御　コンパイラとインタプリタ）／モニタとユーティリティ

第3章　ゲームのプログラムと遊び方……77

サイコロ・ゲーム……78
RNDの使い方／マイコン・サイコロ／丁半ゲーム／画面の更新　CLEAR文／「跳び跳び」運動／CURSOR文
点取りゲーム……90
ゲームの楽しみ方（得点表　2つのサイコロ　勝ちと負け　遊び方）／点取りゲームのプログラム
ムカデ・レース……100

ムカデ(星)の位置／ムカデの動き／ムカデが動けなくなるとき／ムカデの散歩のプログラム／ムカデ・レースのプログラム／プログラムの応用

ブラック・ジャック……112

カード・ゲームのルール／マイコン・ゲームの遊び方／ゲームの手順／プログラムの説明（カードの表示　カードの番号づけ　カードのシャッフル　カードの番号と種類・位との対応）／ブラック・ジャックのプログラム

月着陸ゲーム……128

プログラムの仕組み／プログラムの流れ／プログラムの各部分／ゲームの遊び方／着陸の速度／プログラムの改良

ライフゲーム……139

ゲームのルール／ライフゲームのプログラム（マス目を変数で表わす　プログラム作成上の注意　プログラムの作成）

スタートレック……151

ゲームの内容／ゲームの遊び方／プログラムの説明

囚人のジレンマ……161

マイコンとのジレンマ・ゲーム／マイコンの戦略／ゲームの遊び方／マイコンの学習／ジレンマ・ゲームのプログラム

第4章　実用計算のプログラムと使い方……173

売上げ集計……174

集計／プログラムの手順／ヒストグラム／ヒストグラムの形式／プログラムの作成／最大値を求める／集計とグラフのまとめ／ファイルの使い方（データとアイテム　アイテムとエントリ　エントリとファイル　カセット・テープとファイル　カセット・テープ・ファイルの作成）

在庫問合せ処理……195

在庫ファイル（ファイル設計　ファイルの作成）／在庫捜し／プログラムの使い方

カレンダー……202

曜日を調べるプログラム（曜日を求める計算式　計算するためのプログラム）／ある月のカレンダー（うるう年の求め方　カレンダーの出力形式）／今日を中心としたカレンダー

BASIC FOR MICROCOMPUTERS

第 1 章

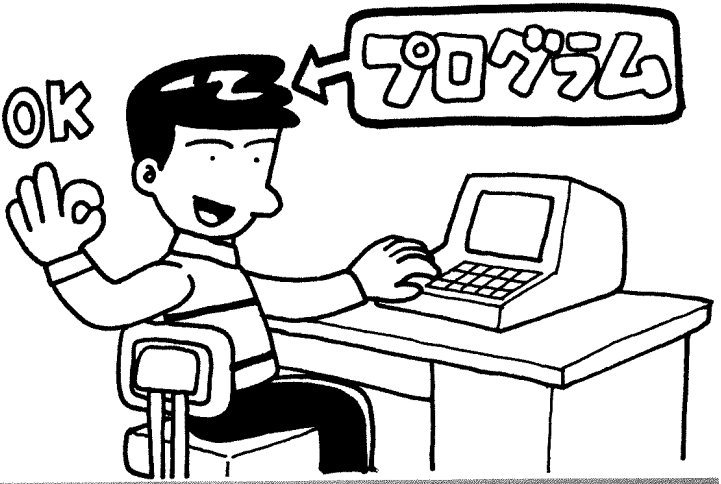
マイコンは、だれにでも使える

マイコンの構造はわからなくても

マイコンの構造はわからないもの

マイコンの中枢部は、いくつかのＬＳＩ（大規模集積回路）です。そしてＬＳＩのそれぞれが、迷路のように複雑な配線で結ばれています。「どうして、ここと、そこがつながっているのか」を、すべて理解するには、エレクトロニクスの知識が必要です。しかし、エレクトロニクスの知識があって、マイコンの内部の配線が理解できても、ＬＳＩの内部までは、とうてい理解できません。マイコンの構造は、一部の専門家を除けば、私たちには「わからないもの」といえましょう。

それでは、マイコンの構造がわからなければ、マイコンは使えないでしょうか。



プログラムがわかれば、マイコンは自由自在に使える

たとえば私たちは、テレビの構造をほとんど知りません。しかし電源のスイッチを入れて、チャンネルを回せば、画像が映ることを知っています。私たちにとって大事なことは、テレビの番組が面白いかどうかで、テレビの内部がどうなっているかではありません。テレビを買うときにも、画面がみやすいか、チャンネルの操作が簡単か、全体のデザインはどうかといったことに注意するはずです。

このような考え方は、内部構造（ハードウェア）を、暗箱（ブラック・ボックス）とみなす考え方といえます。暗箱は、内部が暗いので何もみえませんが、こちらから箱に何かを入れると（インプット）、箱から何かが出てきます（アウトプット）。テレビも一種のブラック・ボックスです。スイッチを入れると（インプット）、画像が映ります（アウトプット）。

このように考えると、構造のまったくわからないマイコンも、テレビなどと同じようにブラック・ボックスとみることができます。

このブラック・ボックスであるマイコンのインプットは、第2章で説明する「プログラム」であり、またアウトプットは、マイコンのすぐれた能力を使って出てくる「仕事」、あるいは「結果」ということになります。

つまり、私たちがマイコンの構造をよく知らなくても、プログラムをよく知っていれば、マイコンは十分に活用することができますし、マイコンもまた、十分に働いてくれます。

マイコンはブラック・ボックス

マイコンの「窓」は周辺機器

マイコンはシステム

マイコンの本体は、いくつかのＬＳＩを複雑に組合せただけの無愛想なものであって、私たちはマイコンの構造はわからないもの、つまり「ブラック・ボックス」としました。では、このブラック・ボックスであるマイコンをどうやって動かしたらよいのでしょうか。

つまり、仕事の指示（インプット）をどこから入れてやり、仕事の結果（アウトプット）をどこから出したらよいのかということです。

このブラック・ボックスであるマイコンへの仕事の指示や仕事の結果を出し入れする「窓」は、いわゆる周辺機器と呼ばれるいくつかの装置ということになります。

一般にマイコンの周辺機器として用いられるものには、

キーボード
ＣＲＴ（ＴＶ）ディスプレイ
プリンタ
カセットテープ・レコーダー
フロッピーディスク

などがあります。

キーボードは、私たちがマイコンに仕事を命令するために使います。マイコンという箱のインプット専用の窓です。

ＣＲＴディスプレイやプリンタを通して、マイコンは、仕事の結果を私たちに報告します。ＣＲＴの画面に表示されたり、プリンタで紙に印字される文字・記号は、マイコンが私たちに伝えるメッセージです。ＣＲＴとプリンタは、マイコン・システムのアウトプットの窓だといえるでしょう。

カセットテープやフロッピーディスクは、インプットの窓にも、アウトプットの窓にもなります。マイコンに命令した仕事や結果を保存しておくために必要な装置です。

これで、内部の仕組みがわからないブラック・ボックスのマイコンにも窓ができました。私たちは、インプットの窓からマイコンに話しかけることができます。またアウトプットの窓からマイコンの答を聞くこともできます。マイコンとの「会話」ができる1つの「システム」になったわけです。

ある装置がシステムであるためには、どこからインプット（入力）し、どこからアウトプット（出力）されるかが、はっきりわかっていなければなりません（図1－1）。LSIを1つ買ってきて、どこに何をつなぐかを考え込んでいる段階では、それはシステムと呼ぶことはでき

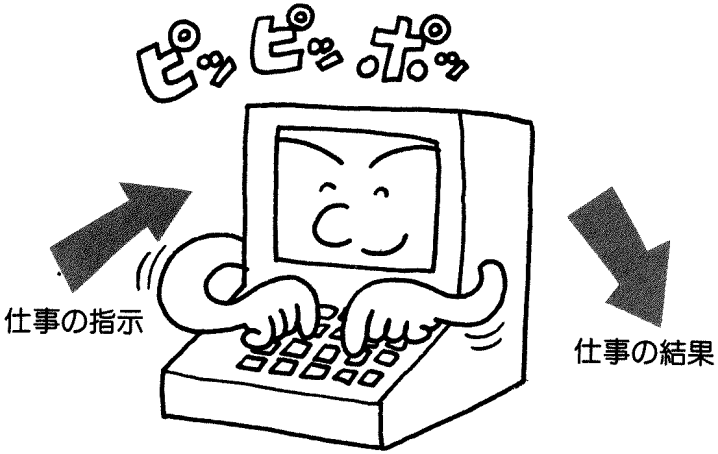
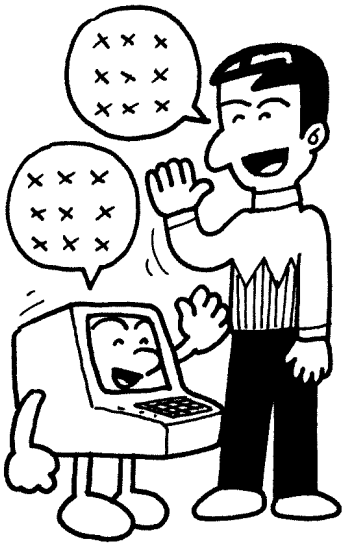


図1－1　マイコン・システム

ません。

私たちは、これから、一通りの周辺機器を備えたマイクロ・コンピュータ・システムと、その活用法を考えていくことにします。人間との「会話」ができること、そして会話に使う「言葉」が十分に簡単なことが、私たちの考えているマイコン・システムの特徴です。わかりやすい言葉で気軽におしゃべりを続けていけば、気難しくて何を考えているかわからなかったマイコンの性格も、だんだんつかめてくるはずです。



システムの実例

「会話が簡単な言葉でできる」という特徴を持ったマイコン・システムは、多くのメーカーから市販されています。これは2つの系統に大別することができます。

ワンボード・マイコン

第1は、ワンボード・マイコンを拡張して、システムを作る形式のものです。初期のマイコン・キットは、基本的な部分だけを、1枚（ワン）の基盤（ボード）の上にまとめたものでした。機能を拡張して使いやすくするためには、ハードウェアそのものを、自分で拡張する必要があります。キーボードや、CRTディスプレイも別に用意しなければならないわけですが、メーカーで、システム拡張用の標準ハードウェア・標準周辺機器を市販している場合が多いようで

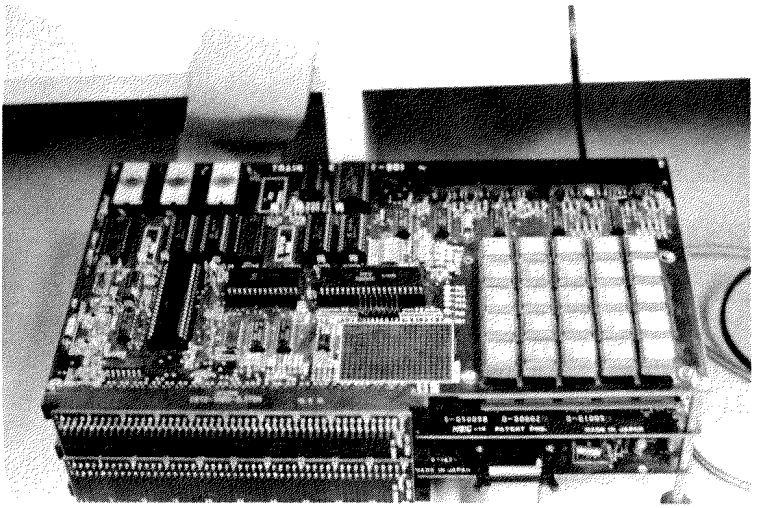


写真１－１　TK－80BS　ベーシックステーション

す。東芝のEX―80BSや、NECのTK―80BS（写真1―1）は、その例です。TK―80BSは、TK―80の基本ボードに市販されている拡張用ボードを増設したもので、専用キーボード、一般用のオーディオ・カセットをコントロールできます。また家庭用テレビに接続し、文字・記号を表示することもできます。

マイコン・セット

第2は「簡単な言葉による会話」ができるように設計されたシステムで、ほとんどが完成品です。キーボードやCRTディスプレイが1つのケースに収まっている場合も多いので、買ってきたその日から使えるわけです。シャープのMZ―80KやコモドールのPET―2001は、カセット装置も内蔵しています。この系統には、パナファコムのPFC―15（写真1―2）のように、ミニコン・レベルの機能と実用性を持っ

たものもあって、気やすく「マイコン」と呼べなくなってきます。

実際にマイコンは、価格的にもオフコン（事務用の小型コンピュータつまりオフィス・コンピュータで、略してオフコンといいます）の小型機と変りがありません。

マイコン、オフコン、ミニコンなどの区別ははっきりしたものではなく、「大きさ」や「価格」、「メーカーでつけた名前」などによって適当に呼び分けているのが現状です。

以上の2つの系統のマイコンは、同じように使うことができます。ただ、これから次々と新製品が登場してきて、さらに使いやすくなっていくのは、第2の系統の「マイコン・セット」といえるでしょう。

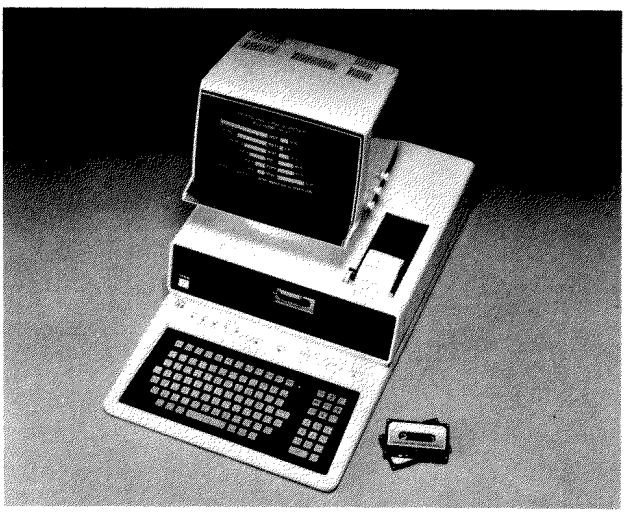


写真1－2　PFC－15

マイコンの周辺機器

周辺機器は、「付属機器」ではありません。それは、私たちがマイコンに語りかけ、マイコンが私たちに答を返す窓口です。しかも、ただ1つの窓口です。マイコンに慣れ、マイコンを使いこなすことは、キーボードを自由に使い、CRTディスプレイやプリンタに思い通りの文字・図形を描かせることから始まります。そして、マイコンの使いやすさのかなりの部分が、これらの機器の使いやすさに依存しています。周辺機器の使いやすさは、キーボードのタッチ、ディスプレイのみやすさ、場合によっては、ケースのデザインをも含むかもしれません。「そんなことより、マイコン本体の性能の方が大切だ」という考え方もあるでしょう。でも、このような考え方からは、マイコンを私たちの生活の中に定着させ、活用しようとする発想は生まれません。

周辺機器は、マイコン・システムの基本です。これから、その1つ1つの構造と機能を調べていくことにしましょう。

CRTディスプレイ

CRTは、カソード・レイ・チューブ（Cathode Ray Tube）の頭文字を取った略語です。直訳すれば「陰極線管」となりますが、特殊なものではなく、普通の家庭用テレビのブラウン管のことです。CRTディスプレイによる文字・記号の表示は、たいていのマイコン・システムで、もっとも重要な出力方法になっています。その意味でも、システムの「顔」だといえるでしょう。

写真1－3　グリーン・モニタ

画面の表示

システムのCRTディスプレイとしては、専用モニタ・テレビを使う場合と、家庭用のテレビを流用する場合とがあります。

文字だけなら、どちらでも、それほど変らない画面になります。ただ、コンピュータ用のモニタ・テレビは、長時間みつめていても、目が疲れないように工夫されていることが多いようです〔たとえば、緑色の螢光面を使ったグリーン・モニタ（写真1—3）など〕。

画面に一度にどれくらい多くの文字を表示できるかは機種によって違いますが、1行32文字で16行か、1行64文字で16行にするのが普通です（20頁。図1—2）。

文字の表示

表示できる文字の種類は、機種ごとに違います。アルファベットの大文字と数字、それに基

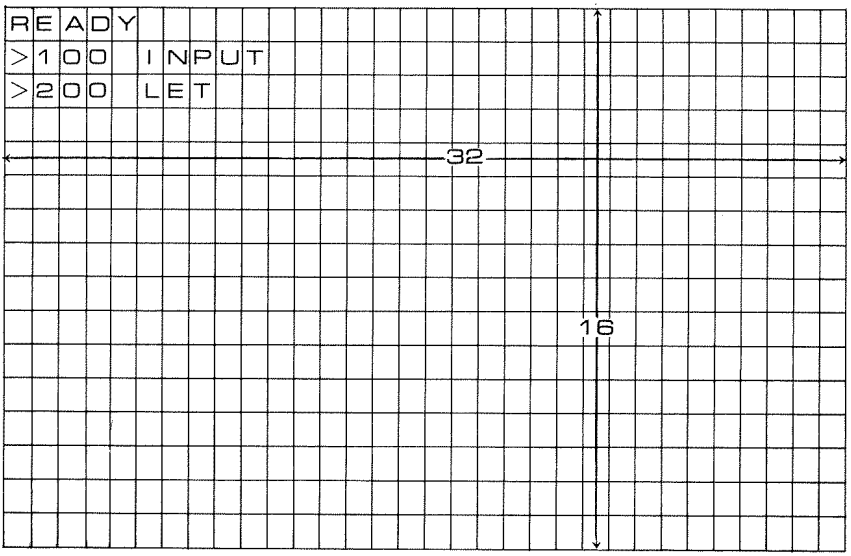


図1－2　画面の大きさ

本的な特殊記号（「＋」、「－」など）は、どの機種でも使えるはずです。カナを表示できる機種も多く、TK—80BSでは、英数字、カナを含めて255種類もの文字・記号を表示できるようになっています。

ドット方式

画面に表示された文字をよく調べてみると、小さな点がいくつか集まって、文字の形になっていることがわかります。これを、ドット方式といいます。ドットの細かさは、5列7行（5×7）と、7列9行（7×9）の二通りがあります（図1—3）。ドット方式はCRTにかぎらず、プリンタなどにも用いられる方法です。

キーボード

電卓なら、数字キーが10個（テンキーといいます）と、演算・関数キーがいくつかあれば、

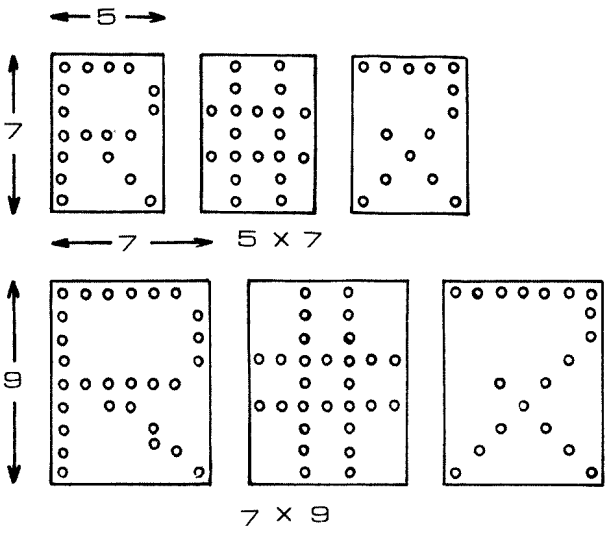


図1－3　ドット方式

立派なキーボードになるでしょう。でも、マイコンとの「会話」を楽しむためには、さらに豊富な機能が必要です。それでは、マイコンに対する命令を1つずつのキーに対応させて、「このキーを押したときは、この命令」と知らせる方法をとれば、それで豊富な機能を実現できるのでしょうか。プログラム電卓は、「1つの命令に、1つのキーを対応させる」という、この方法をとっていますが、マイコン・システムでは、も

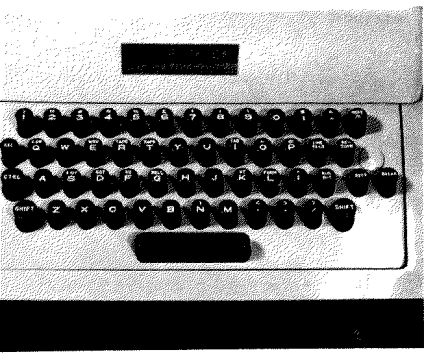


写真1－4　キーボード

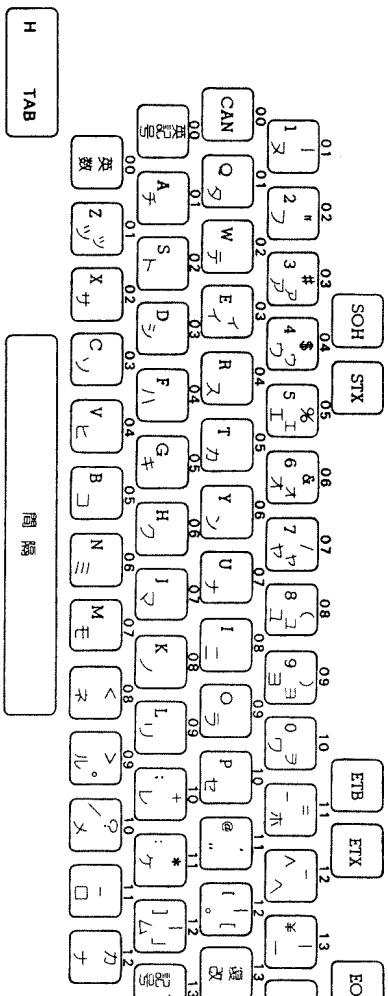


図1－4　キーボード配列

っと一般的なやり方を採用しています。

それは、マイコンへの命令を、人間の言葉に似た言葉で表現することです。「計算しろ」とか、「画面に出せ」という形式です。ところがコンピュータを初めて作った人が、たまたま英語を話す人々だったために、「人間の言葉に似た言葉」は、「英語に似た言葉」になってしまいました。

「計算しろ」は、例えば、「COMPUTE」という英語で、マイコンに伝えられます。私たちは、キーボードのキーを

C O M P U T E

の順で、押していけばよいのです。

こういう英語の文章を伝えるには、英文タイ

プライタと同じ形式のキーボードを使うのが便利です。慣れないと使いにくいかもしれませんが、文字・記号の数が、日本語よりずっと少ないので、すぐに覚えられます。それに、第2章で説明する「ベーシック」という言葉には、難しい英語が出てきません。いくつかの基本的な単語（キーワードといいます）だけで、文章を作るようになっています。

タイプライタは、キーを押すと、それに対応した活字を印字します。マイコンのキーボードでは、キーを押すと、その文字をCRTディスプレイの画面に出すことになっています。私たちは、キーボードとCRTディスプレイを一種のテレビ・タイプライタのように使うことができます。

JISとASCII

キーボード上のキーの配列（図1—4）には、JIS（ジス）とASCII（アスキー）の2種類があります。JISキーボードには、カナ文字のキーがついていますが、ASCIIにはついていません。ファンクション・キーといわれるキーの配列も少し違います。また、同じJIS、ASCII規格でも、メーカーによって、いくらか異なることがあります。

キーボードの使い方

ここで、TK—80BSのベーシックで、プログラムを作ってみましょう。プログラムの意味は、第2章までおあずけにします。キーボードの扱い方にだけ慣れてください。

TK—80BSの専用キーボードは、JIS規格ですが、

- 「英数」のキーがなく、その場所に「BREAK(割込み)」キーがある。
- 「カナ」キーだけがロックされる。

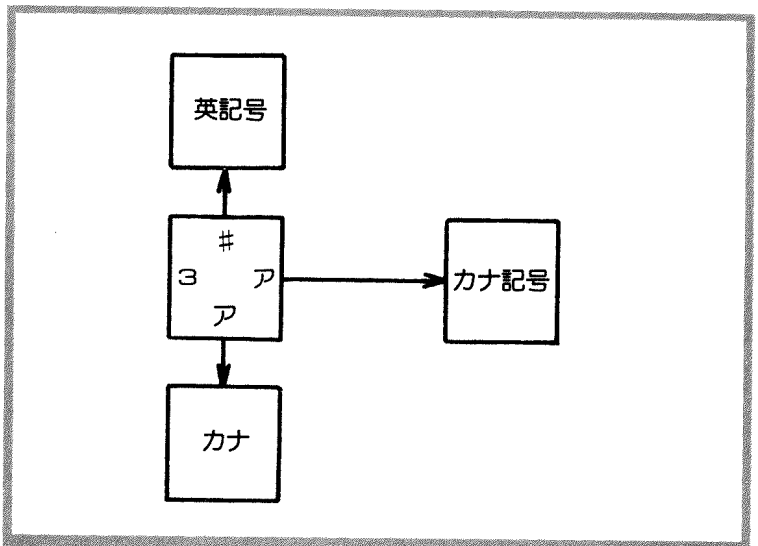


図1－5　シフト・キーの使い方

という点が違います。「BREAK」キーの役割は、第2章で説明します。「ロックされる」というのは1回押しただけで、ずっと押し続けているのと同じ状態になるという意味です。「英記号」「カナ」「カナ記号」キーの3つをシフト・キーといいます。

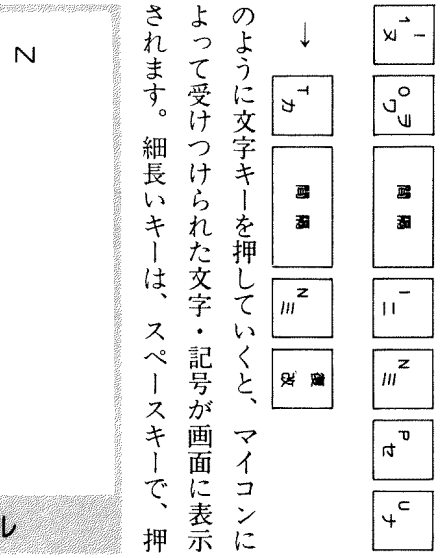
シフト・キーの使い方

文字キーのキートップ（指で押す部分）には、文字・記号が書いてあります。シフト・キーを押さない状態で文字キーを押したときは、左端に書かれている文字を入力したことになります。たとえば図1－5では「3」です。図1－5で「＃」「ア」「ア」を入力するためには、それぞれ「英記号」「カナ」「カナ記号」のシフト・キーを押しながら文字キーを押します。

文章の入力

```
10 INPUT N
```

と、入力するために、



のように文字キーを押していくと、マイコンによって受けつけられた文字・記号が画面に表示されます。細長いキーは、スペースキーで、押すと次の文字の前に1文字分の空白ができます。空白をたくさん作りたいときには、何回も押せばよいわけです。

```
>10  INPUT  N
>
   ↑
  カーソル
```

図1－6　カーソル

「復改」キー

最後に押す「復改」のキーは「1行が終りました」と知らせるためのもので、カーソル（今キーを押すと、この位置に文字が表示されると教える記号で、いつも図1－6のように画面に出ています）が、次の行の先頭に移動するはずです。ベーシックでは、行の先頭に「>」の記号が出ることになっていて、カーソルが次の行に移るときに、マイコンが勝手に表示します。

「後退」キー

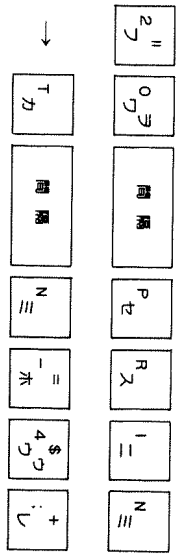
キーの押し方を間違えたときは、「後退」のキーを押すと、カーソルが1文字分戻ります。そのあとで、正しいキーを押せば、押したキーの文字が入力されます。

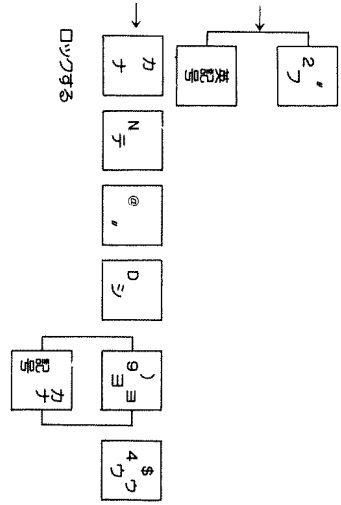
「まつ消」キー

1行全部を書き直したいときは「まっ消」のキーを押してください。カーソルが行の始めに戻りますから、改めて、入力していくことができます。ただ、「復改」キーを押してしまったあとで、この方法を使うことはできません。「復改」キーは、マイコンに「今までの入力が1行で、これから、次の行を入力する」ということを通知するのです。

さらに続けてみましょう。

20 PRINT N−4;"テ゛ショウ"と入力します。





同様にして、30 END を入力すればプログラムの入力は終りです。

プログラムを作っていく段階では、私たちが入力した文字が、そのままCRTの画面に表示されていくだけですが、マイコンは、私たちの

```
>10  INPUT  N
>20  PRINT  N−4;"ﾃﾞｼｮｳ"
>30  END
>
```

図1－7　プログラム入力の画面

指示を全部覚えています。画面を消してしまっても、プログラムが消えることはありません。

画面に出てくる文字（図1―7）は、一度、マイコンを通ってきたものです。私たちがキーを押した証拠ではなく、マイコンが入力を受けつけて、しっかりと覚えたという証拠です。この点にだけ注意しておいてください。

マイコンの答

マイコンがどのように、私たちの指示にしたがってくれるのかをみるには、マイコンにプログラムを実行させなければなりません。

そのためには、

RUN

と入力します。「?」が画面に出ますから、適当な数、たとえば、

47

と入力してやると、図1―8のように出力しま

```
>RUN
?47
  43  テ゛ショウ
>RUN
?16
  12  テ゛ショウ
```

図1－8　実行どきの画面

す。

枠で囲んだ部分が、プログラム実行の結果です。マイコンは、入力した数から4を引いた数を出力してくるという、奇妙な答え方をします。

「なぜそうなのか」は、第2章まで待ってください。

プリンタ

CRTディスプレイの画面は次々に書き換えられていきます。表示するのに必要な時間が短かく、スピーディなのは便利です。でも、画面を切り取ってポケットに入れておくということができませんし、翌日まで画面を保存しておくのも難しいでしょう。プリンタは、プログラムや、プログラムの実行結果を、目にみえる形で保存しておきたいという場合には、もっとも必要な装置です。

プリンタの種類

プリンタには、機器組込み用のものから、高級なライン・プリンタまで様々な種類のものがあります。マイコンの周辺機器としてよく使われるのは、ドット・インパクト型のものです。CRTの項で説明しましたが、5×7、あるい

は7×9のドットを使って文字・記号を印字します。インクを使わないノン・インパクト型には、サーマル・プリンタや放電プリンタがあります。どちらも特殊な処理を施した用紙が必要です。

サーマルプリンタ

サーマル・プリンタ（ドット方式での印刷・

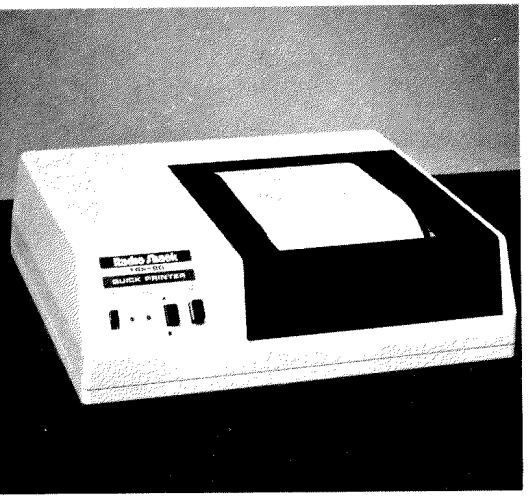


写真1－5　プリンタ

写真1－6）では、独特の青い色の印字が得られます。用紙にタバコの火を近づけると青く変色するのでわかりますが、ドットを印字するのに熱を加えるのです。これらのノン・インパクト型は、マイコン用にも普及しています。無音ですし、高速印字が可能です。

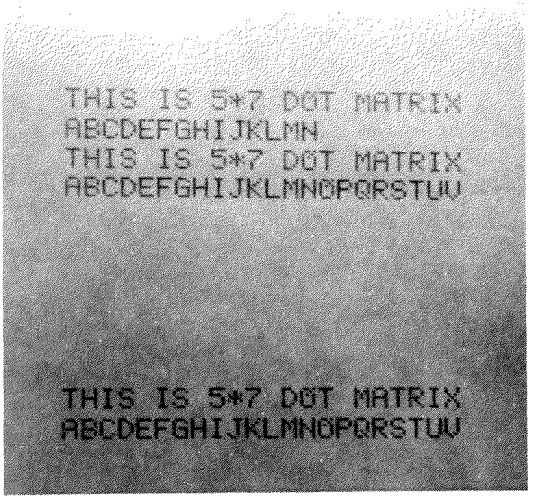


写真1－6　ドット方式の印刷

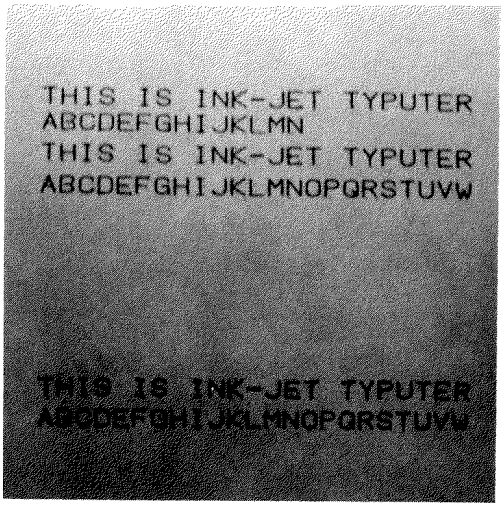


写真１－７　インクジェット方式の印刷

インクジェット方式

ほかに、インクジェット方式（写真１―７）といってインクを文字の形に吹きつけるものもあります。印字中のノズルの前に指を出すと、指にも、きれいに印字してくれます。まだ、マイコン用には普及していません。

これらのプリンタは、プログラムからコントロールされるだけでなく、ＣＲＴの画面のハードコピー（画面に出ている文字・記号を、そのまま紙に写すこと）を取るためにも使われます。ハードコピーをきれいに取るためには、もちろん、画面の１行に表示される文字の数より、プリンタの１行に印刷できる文字の数の方が多くなければなりません。

カセット・テープ

プログラムやデータ（マイコンに処理させたい資料）は、プリンタなどを使えば、目にみえる形で保存しておくことができます。コピーを取ることもできますし、他人にみてもらうこともできるでしょう。ただ、同じプログラムを、もう一度マイコンに実行させたい場合や、同じ

データを、違うプログラムで処理したいという場合は、また、キーボードから入力してやらなければなりません。マイコンにプログラムが印刷された紙を見せても、何もしてくれません。

何度も使うプログラムやデータを、私たちにわかりやすい形式で保存するのと別に、マイコンにもう一度入力できる形式で保存しておくことができないでしょうか？

カセット・テープや、次の項で説明するフロッピーディスクは、そのための装置です。マイコンの周辺機器として使うカセットテープは、一般のオーディオ・カセットで十分です。

オーディオ・カセットとマイコンとの接続の仕方には、カンサス・シティ・スタンダードと呼ばれる規格が定められています。

写真１－８　カセット装置

フロッピーディスク

フロッピーディスク（写真1―9）やミニ・フロッピーディスクは、やや高級なマイコンに使われます。オーディオ・カセットよりたくさんのデータを保存することができますし、データを読み出し（フロッピーからマイコンへ）たり、書き入れ（マイコンからフロッピーへ）たりする操作の効率がよいからです。

それは、なぜでしょう？

たとえば、カセットテープの始めから2mのところに読み出したいデータが入っているとします。いくら場所がわかっていても、やはり、テープを始めからたどっていって、2mの位置をみつけなければなりません（シーケンシャル・アクセス図1―9①）。テープを巻くには時

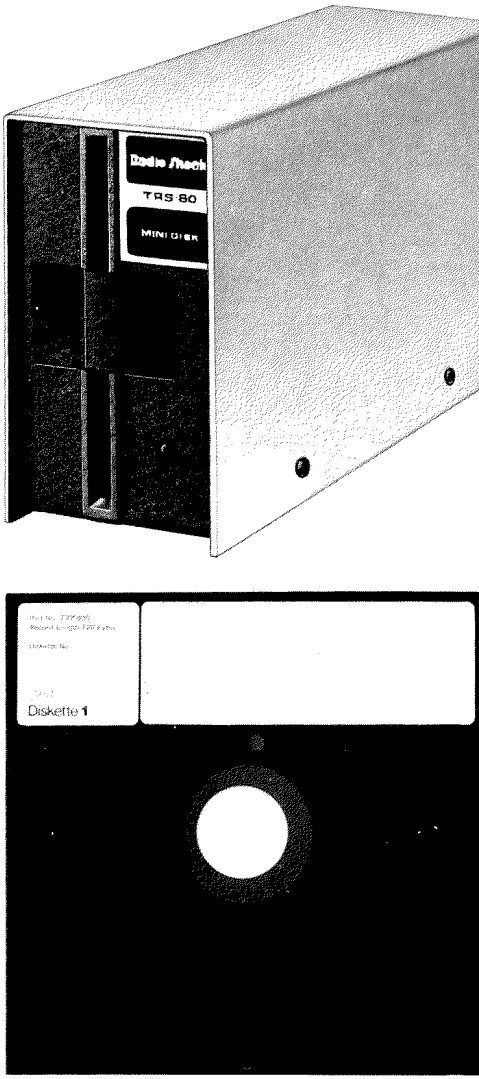


写真1－9　フロッピー装置/TRS(上)とフロッピーディスク(下)

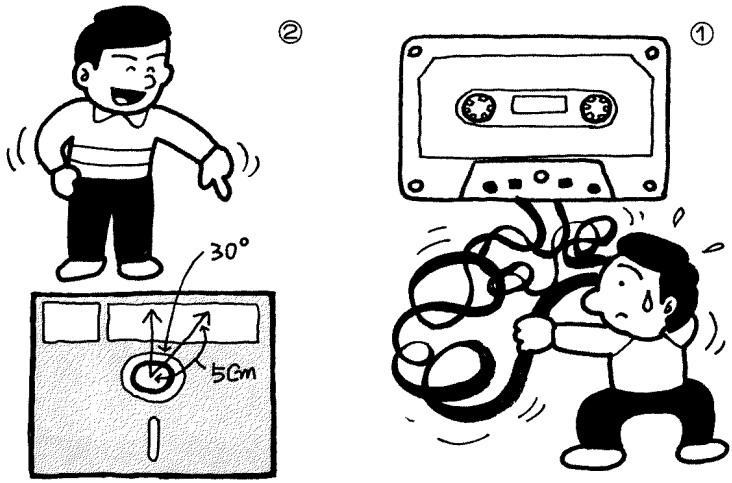


図1－9　アクセス

間がかかりますから、テープの後の方にあるデータほどみつけるのが遅くなります。

これに対して、フロッピーディスクでは、ディスクの中心から何cm、基準線から何度と、データの位置を指定すれば、データがどこにあっても、だいたい同じ時間でみつける(アクセスする)ことができます(ランダム・アクセス図1―9②)。これが、フロッピーディスクの利点です。大量のデータを一時に読んだり、書いたりするときは、テープでもディスクでも、それほど効率の違いはありません。比較的少ないデータを、何回もマイコンとやりとりする場合は、ディスクの方がはるかにすぐれています。

カセットテープやフロッピーディスクを使えば、プログラムやデータを、いつでもマイコンに渡せる形で保存しておくことができますし、ワンタッチで、マイコンに渡してしまうことも

できます。これらの装置は、「マイコンの実用化」に大きな役割を果しました。

写真1－10　ASR－33タイプライタ

そのほかの周辺機器

マイコンの周辺機器というより、コンピュータ一般の周辺機器として用いられるものを、いくつか説明します。

タイプライタ

キーボードとプリンタを合わせたものです。むしろ、キーボードとプリンタが、タイプライタを分けたものというべきかもしれません。コンピュータの周辺機器としては、もっとも古いものに属します。もちろん、手動のタイプライタをマイコンにつなごうとしてもだめです。「端末用」の機能を持ったものでなければなりません（周辺機器は、コンピュータと人間との接点ですが、コンピュータの側からみれば中心から一番遠い「端末」です。「周辺機器」も「端末」も、いずれも、視点を機械の側に定めた言葉といえるでしょう）。

タイプライタのベストセラーはASR—33（写真1—10）と呼ばれる機種です。一昔前は、コンピュータのあるところには、必ずこのASR

—33がありました。安価で丈夫ですが、印字速度がきわめて遅く、騒音が大きすぎるのが難点です。

タイプライタには、ほかに、IBMのセレクトリック・タイプライタのような卓上型のものもあります。中古品（リースバック品）は、意外に容易にみつかります。

紙テープ・リーダ、パンチャー

紙テープ（写真1—11）は、かなり長い間、重要な外部記憶媒体でした。外部記憶媒体というのは、マイコンの外にデータを保存しておくものです。カセットテープやフロッピーディスクも外部記憶媒体です。紙テープは、一度穴をあけてしまうと、修正がききません。ただ、装置も手軽ですから、マイコン用の周辺機器としては、まだ使えそうです。

紙テープには8単位テープと6単位テープがあります。ASCII、EBCDICの文字コ

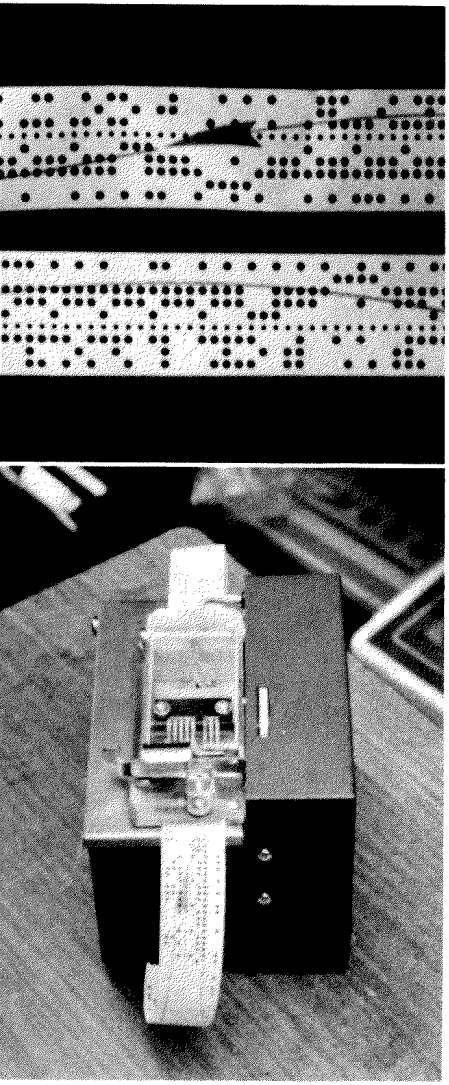


写真1—11　紙テープ(上)と紙テープ・リーダ

ードを、そのままさん孔（穴あけ）できる8単位テープが使いやすいでしょう。タイプライタのASR—33などには、紙テープ・リーダ、パンチャーの機構が付属しています。

XYプロッタ

マイコン用として、これから有望なのは、むしろ、XYプロッタなどの出力機器でしょう。XYプロッタというのは、印刷用紙上にペンを置く位置をXY座標（中心から上に何cm、左に何cmというように）で表わして、曲線や図形を描くものです。もちろん、文字を書くこともできます。統計処理の結果をグラフ化して、そのまま報告書などに転載する活用の仕方などもあるでしょう。

以上のように、多彩な周辺装置が、マイコンの周辺機器として使えます。これらの機器の中には、マイコン本体よりも大型で高価なものもあります。

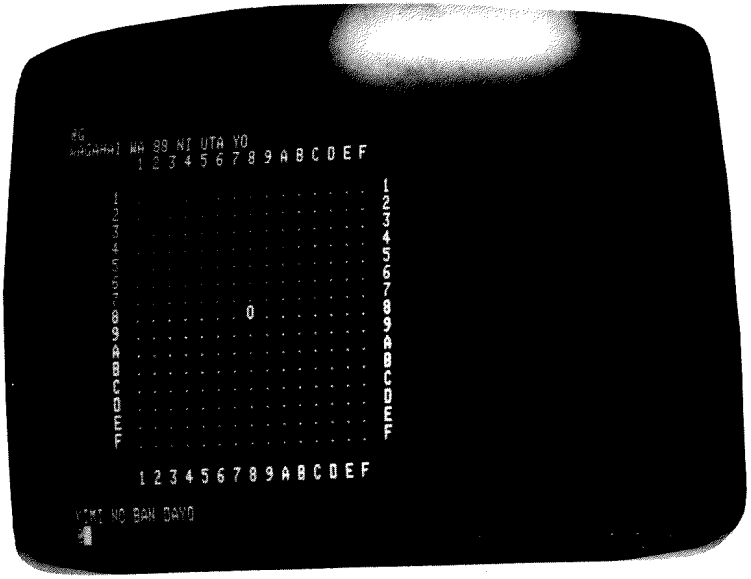
マイコンを使う私たちの側からすれば、重要なのはマイコン本体よりも、これらの「周辺機器」であり、マイコンが、私たちのために本当に役立ってくれるかどうかということは、これらの周辺機器がどれだけ活用されるかにかかっているといえます。

もちろん、マイコン本体が不要だといっているわけではありません。マイコンは、これらの周辺機器を使いこなす言葉を私たちに提供してくれます。周辺機器を私たちの思い通りに働かせるための言葉が「ベーシック」と呼ばれる魅力的な言葉です。次の章で、この「ベーシック」を用いてマイコン・システムを使いこなす方法を、考えていくことにしましょう。

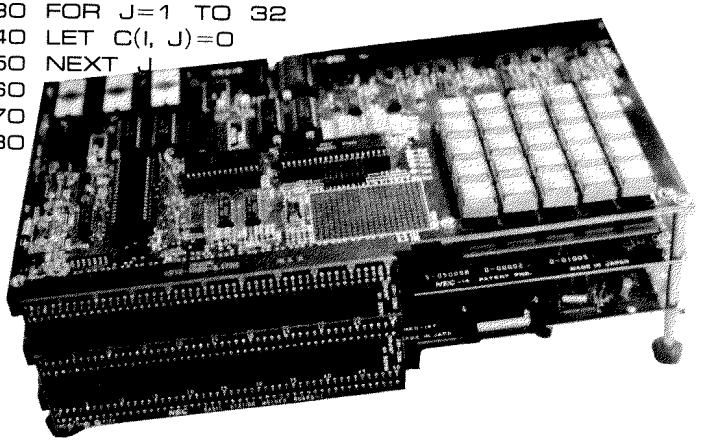
BASIC FOR MICROCOMPUTERS

第 2 章

ベーシックによるプログラミング



```
*FS301R4 STOP AT 495. END OR CONT -
 END
*OLD
*FS101R4 OLD PROGRAM NAME-KENT
*LIST
   100  REM  SPIRAL
   110  DIM  C(16, 32)
   120  FOR  I=1  TO  16
   130  FOR  J=1  TO  32
   140  LET  C(I, J)=0
   150  NEXT  J
   160
   170
   180
```



マイコンとプログラム

コンピュータは、とても優れた能力を持つ機械です。でも機械ですから、私たちが何かをするように命令しないと動いてくれません。また誤った使い方をすると、とても大変なことがおこるかもしれません。

それはあたかも、自動車の運転の仕方を知らない人が、高速道路を時速百キロでとばすようなものです。

車を安全に運転するためには、それに必要な知識を学ばなくてはなりません。私たちがマイコンを使うためには、マイコンにどのような命令を与え、どのように仕事をさせるのかを知ることが必要です。

この章では、マイコンを使うための知識、つ

まりプログラムとは何か、またプログラムはどのように書くか、などについて説明しましょう。

プログラムとは

プログラムとは、マイコンにどのように仕事をさせるか、またマイコンにどのように命令するかを、マイコンにわかる言葉で記述したものです。

私たち人間は、歩きながら話をしたり、食べ物を食べながら本を読んだりすることができます。しかし、マイコンは機械であって、一度に1つのことしかすることができません。ですから、まず何をやり、次に何をやるかを、詳しく命令してやらないと、マイコンは何にもできないわけです。

このマイコンがやるべき仕事を詳しく順序だてて書いたものが、プログラムというものです。

それでは、ただ順序よく仕事が並んでいれば、マイコンのプログラムになるでしょうか。

私たちは試験の前などに、よく予定を組みます。まず国語の勉強をして、それが終ったら数学をやるなど。これも1つのプログラムといえるでしょう。でも、このような大ざっぱなプログラムをマイコンに与えても、マイコンは働いてくれません。

それは、マイコンがコンピュータといえども、やはり機械であって、自分で物事を勝手に判断して、適当にやっていくということを知らないからです。

したがって、マイコンに与えるプログラムは、あいまいさがなく明確で、細かいところまで指示されたものでなくてはならないものといえます。

マイコンのプログラムが持つべき条件をまと

めてみると、次の3つのことが考えられます。

1　順序よく仕事が並んでいること

2　あいまいさがなく十分に細かく指示されていること

3　マイコンにわかる言葉で書かれていること

したがって、マイコンのプログラムを書くためには、この3つのことを学ばなければなりません。

ここでは、1と2のために、プログラミング（プログラムを書くこと）の方法の1つとして、フローチャート（流れ図）を書くことを学び、3のマイコンにわかる言葉として、BASIC（ベーシック）というコンピュータの言語を学んでいくことにしましょう。

プログラムの流れ

前述したように、プログラムを書くためには、やるべき仕事を順序よく並べていかなければなりません。このようにいうと、たとえば、朝起きて、食事をして、学校にいって、帰ってきて、テレビをみて、勉強して、寝る、このように、やるべき仕事が並んでいることを思い浮べるのではないでしょうか。

たしかに、これらのことは、順序よく並んでいます。しかし、これらのことは、あいまいさがなく、十分に細かく指示されているといえるでしょうか。

たとえば、3番目の学校に行くことだけをとりあげてみましょう。私たちは、ほとんど毎日学校に通っていますから、学校に行くという言葉だけで、その言葉が意味すること（何という

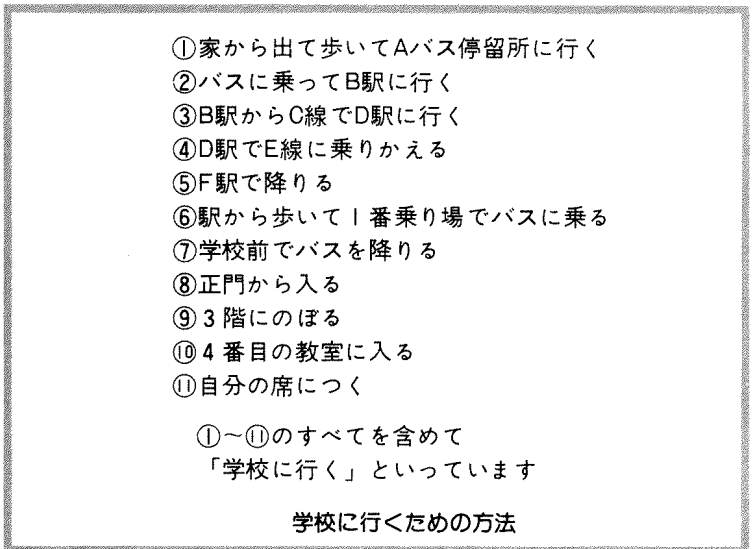


図2－1　フローチャートを書く手順

学校へ行くのか)、その方法（どのような道を通っていくのか)、それに要する時間（どれくらいかかるか）が、ぱっと頭に浮んできます。しかし、あなたがその学校を知らない人に、学校のことを説明するときのことを考えて下さい。

学校に行く方法だけでも、図2—1のように、たくさんのことを説明しなければならないでしょう。私たちは、もう毎日慣れてしまっているので気がつきませんが、無意識のうちに図2—1のようなことを行っているわけなのです。

マイコンは、ここでは見知らぬ他人のようなものです。だから、私たちはマイコンに仕事をさせるときには、本当に細かいことまで十分に記述しておかなければなりません。そして、そのように記述するときの手助けとして、仕事の手順（これを流れ—フローという）を図に表わす技術が開発されています。この方法を使うこ

とによって、私たちは、プログラムを誤りなく作っていくことができるようになります。このようにして作られた図を、流れ図（フローチャート）といいます。

フローチャートの書き方

フローチャートの簡単な例を図2—2に示します。

図2—2は、おなかがすいたときに、ラーメン屋に入ってラーメンを食べる。その手順を書いたフローチャートです。どういうことが書かれているかは、だいたいわかると思います。

まずラーメン屋に入ります。そして、ラーメンを注文し食べます。1杯食べたところで、おなかがいっぱいになれば、金を払って店を出ます。まだ、いっぱいでなければ、もう1杯ラーメンを食べます。

このような仕事の手順を詳しく図で示したも

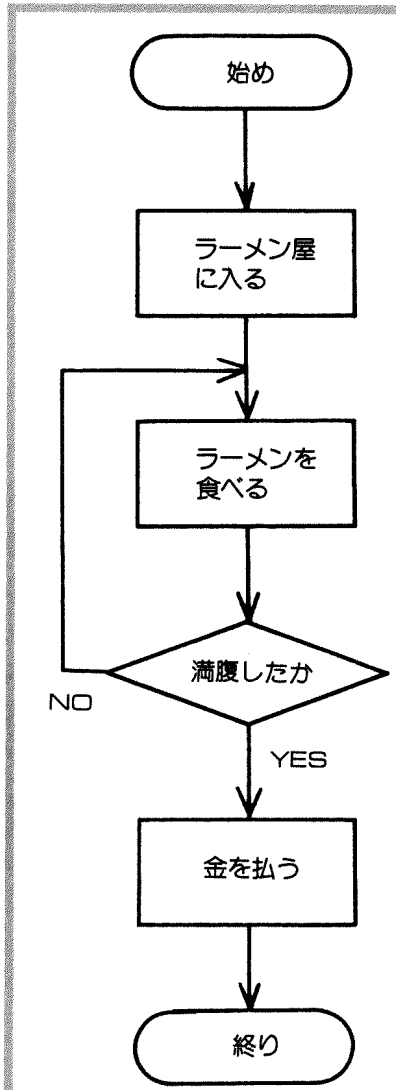


図2－2　フローチャート

のをフローチャートと呼び、これにそってプログラムを作って行くのです。

まず、フローチャートを書くための記号をいくつか学びましょう。

(1) ターミナル

ターミナル記号とは、図2—3の上と下にある始めと終りの記号（だ円形のもの）です。

プログラムには、常に始まりと終りがあります。ターミナルとは「端」の意味で、プログラムの両端を示しています。フローチャートの最初と最後には、必ずこの記号が書かれます。

(2) 各種処理

図2—4の長方形の図形が各種処理を表わします。各種処理とは、仕事を細かく分けたときの、その1つの部分を表わします。したがって、どのくらい細かく仕事を分けるかによって、各種処理の記号が多くなったり、少なくなったり

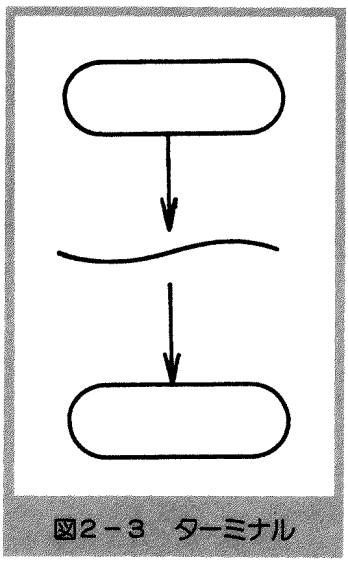


図2－3　ターミナル

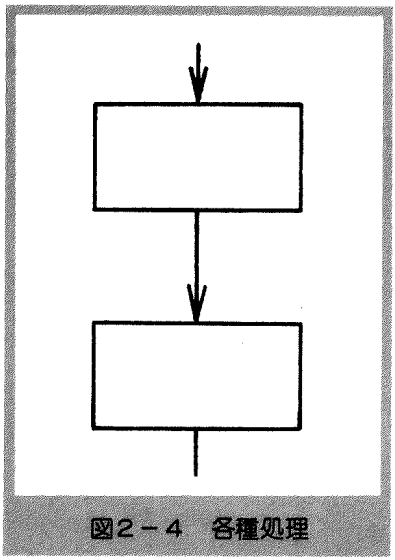


図2－4　各種処理

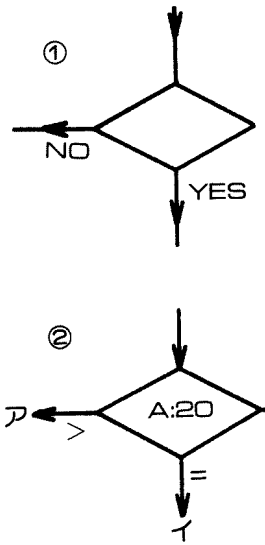


図2－5　判断

します（つまり、図2―2の「ラーメン屋に入る」という処理は、「ラーメン屋をみつける」・「ラーメン屋の入口をあける」・「入る」・「いすにすわる」などに分けることもできます）。

ただ、あまり仕事を細かく分けすぎると、かえって複雑になるので、適当に、あいまいにならない程度に区分していくとよいでしょう。

(3) 判断

私たちの仕事、また、マイコンの仕事には、判断はつきものです。その判断を表わす記号は図2―5①のひし形の記号です。ここでの例のように、ひし形の中に書かれてある文章（満腹したか）に対する答がＹＥＳであれば、その矢印の方向に、ＮＯであれば、その矢印の方向にそれぞれ移ります。

判断の記号は、また図2―5②のように書くこともあります。これはＡが20より大きいときには、アの方へ行け、Ａが20に等しいときには、イに行け、Ａが20より小さいときには、ウに行けという意味です。

このように、判断の記号のところでは、道が2つ、あるいは、それ以上にわかれます。これは、各種処理やターミナルと、大きく異なる点です。

(4) 流れ線

いままで出てきたターミナル記号、各種処理

記号、判断記号などの間をつなぐ矢印です。プログラムは、この矢の向きに向かって流れて行きます（図2―6）。

フローチャートで使われる記号には、まだたくさんありますが、マイコンで使われるものとしては、あと、コネクタ、定義済処理（サブルーチン）があります。

⑸ コネクタ

長いプログラムのフローチャートになると、ノート1ページの中におさまりません。そこで、図2―7のようなコネクタを使って、適当なところ（あとでみやすいように）で、フローチャートを切り、分割します。

したがって、コネクタは、フローチャートの部分的な出入口になります。丸の中には、何らかの文字をつけて、どこの出口と、どこの入口がつながっているのかを、わかりやすくしておくとよいでしょう。

⑹ 定義済処理

定義済処理（図2―8）は、長いプログラム

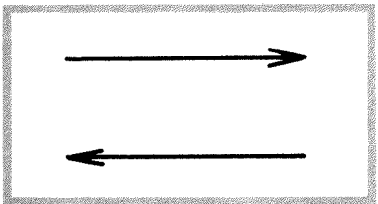


図2－6　流れ線

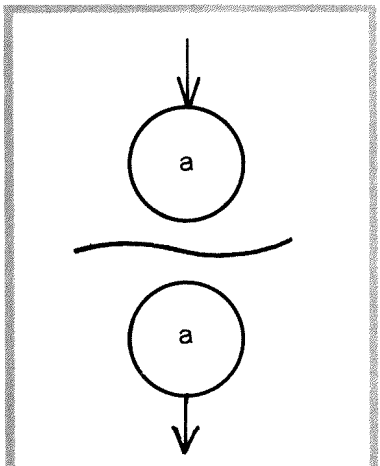


図2－7　コネクタ

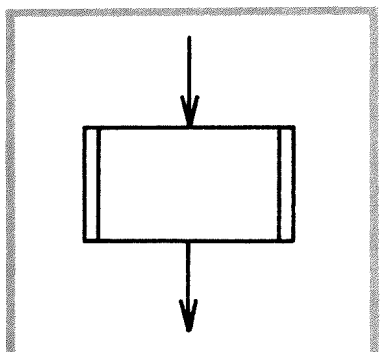


図2－8　定義済処理

のフローチャートなどでよく使われます。あるフローチャートの中で、ほかのフローチャート（すなわち、ほかのプログラム）によって定義されているものを使うときに、この記号で示します。たとえば、ある人の1日をフローチャートで書くとしたとき、その人が毎食、ラーメン屋でラーメンを食べているとしたら、図2—2を定義済処理として使って、図2—9のように書きます。

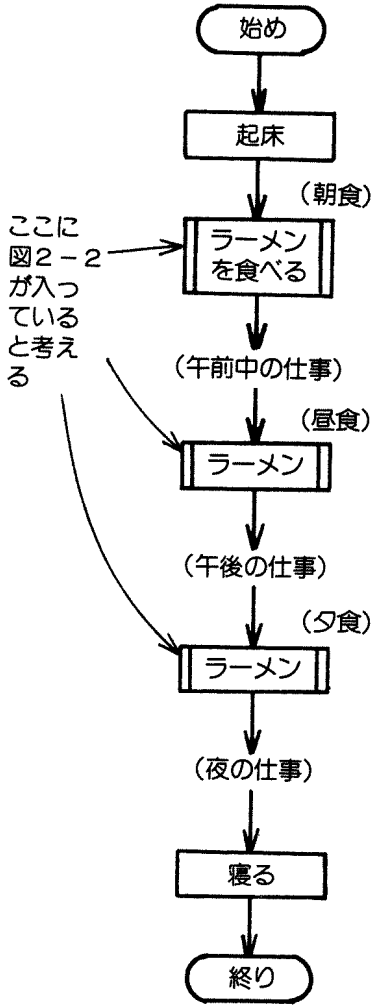


図2－9　1日の生活のフローチャート

以上の6つの記号を使えば、マイコンのプログラムとしては、ほぼ十分なくらいのフローチャートを書くことができます。しかし、このフローチャートは、プログラムではありません。マイコンは、フローチャートを読むことはできません。フローチャートは、単にプログラムを書くための手助けなのです。私たちは、フローチャートから、マイコンが理解できる言葉を使ってプログラムを書くわけです。

ベーシックとは

マイコンは、人間の言葉をそのまま理解してはくれません。私たちが、マイコンに命令を与えるときには、マイコンに理解できる明瞭な言葉を使わなければなりません。

プログラム言語の種類

その目的のために作られた特殊な言葉をプログラム言語といいます。人間の言葉に、日本語、英語、スペイン語など多くの種類があるように、プログラム言語にも、実に多くの種類があります。現在、よく使われているものだけでも、

アセンブリ言語
フォートラン（FORTRAN）
コボル（COBOL）
アルゴル（ALGOL）
リスプ（LISP）
ベーシック（BASIC）
PL／I
APL

などをあげることができます。

なぜ、ベーシックを使うか

これだけ多くのプログラム言語の中で、なぜベーシックを取りあげて学ぶのでしょうか。その疑問に対する答は、「ベーシックが一番簡単だから」です。ベーシックより簡単で使いやすいプログラム言語は、今のところ作られていません。

このことは、文章で説明するよりも、実例をみてもらった方が、わかりやすいでしょう。同じ意味のことを、いくつかの言語で表現してみると、図2—10のようになります。

日　本　語　2つの数A,Bを入力して、その和A+Bを出力せよ

ベーシック
```
100 INPUT A,B
110 PRINT A+B
```

フォートラン
```
      READ (5,100) A, B
100   FORMAT (2I3)
      WRITE (6,200) A+B
200   FORMAT (' ',I4)
```

リ　ス　プ
```
DEFINE(((PAB(LAMBDA (A B)
                (PLUS A B)) )))
PAB((READ) (READ))
```

アセンブリ言語
```
        BSS  INPUT
        ST   R0, A
        BSS  INPUT
        ST   R0, B
        LD   R1, A
        AD   R1, B
        MOV R0, R1
        BSS  OUTPUT
```

パスカル
```
VAR A, B : INTEGER;
  BEGIN READ (A, B);
        WRITE (A+B) END.
```

機械語(16進)
```
8FE7
47C5
8FE5
47C4
C325
D102
50B2
8FE0
```

機械語（2進）
```
1000111111110011
0100011111000101
1000111111100101
0100011111000100
1100001100100101
1101000100000010
0101000010110010
1000111111100000
```

図2－10　プログラム言語のいろいろ

ベーシックが、少なくとも、とっつきよさそうな言語だということはみてとれるでしょう。

そこで、本書では、機械語、アセンブリ言語でプログラムを組まないで、ベーシックでプログラムを組むことにしました。

現在、市販されているマイコンは、ほとんどベーシックが使えます。ただ、メーカーによって規格が違う点もあり、全く同じベーシックの使い方ができるというわけではありません。

ベーシックの国際的規格は、ＡＮＳＩなどで定められていますが、本書では、どんなマイコンでも、必ず使える文や単語を集めた規格として、ＭＩＮＩＭＡＬ（最小）ベーシックの規格を採用しました。

ですから、本書に掲載されているプログラムは、ほとんどのマイコンに使うことができるはずです。

ベーシックの使い方

ベーシックを使うためには、それなりの手順があります。

まず、マイコンに電源を入れなければならないのは、当然です。

電源の入ったマイコンは、キーボードからの入力を受けつけることのできる状態になっているはずです（「スタート」ボタンなどを押さなければならない場合もあります）。

ここで、「これからベーシックを使いますよ」という意味の命令、たとえば、「ＢＡＳＩＣ」という文字を、キーボードから入力します。

マイコンが、この命令を理解すると、「ベーシックが使えます」という意味の表示が、画面に出ます。

これで、ベーシックが使えるようになるわけ

です。

たとえば、TK—80BSのベーシックでは、図2—11のような手順で行います。

図2—11で、下線を引いてある部分は、私たちがキーボードから入力した文字が、そのまま、画面に出力されたものです。下線の引いていない部分は、私たちが入力したものではなく、マイコンの方で出してくれるメッセージです。

```
*BASIC STATION
      MONITOR V1.0*
$ BA
RAM END? 97FF
*LEVEL-2 BASIC
                V1.0*
>100 PRINT "!"
>
```

図2—11　TK80BSのベーシックの使い方

下線を引いた文字のうち、「BA」という文字が、マイコンに、ベーシックを使いたいということを通知しているのです。

最後の行の「>」は、促進記号（プロンプト記号）と呼ばれ、マイコンが、私たちに入力を要求していることを示しています。

このプロンプトの後に、ベーシックのプログラムを入力していくわけです。

ベーシックのプログラム

図2—12は、2つの数を入力して、その和を求めるプログラムと、その実行結果です。この計算自体は、単純なことで、電卓なら、一発で答を出すでしょうが、ベーシックに慣れていないみなさんには、まだ、わかりにくいかもしれ

① >10 INPUT A,B

② >20 LET C=A+B

③ >30 PRINT A; "+" ;B; "=" ;C

④ >40 STOP

⑤ >RUN

⑥ A? 10

⑦ B? 3

⑧ 10+3=13

⑨ >

それぞれの文の意味

①2つの数をキーから入力し、それをA,Bの2つの変数に入れる。

②AとBを足して、その値をCに入れる。

これはA+BとCが等しいという意味ではなくAとBの和をCという変数に与えることを意味する。

フローチャートのように、C←A+Bの方がよい表わし方だが、慣習的にC=A+Bとしている。

③Aの値と+という文字、Bの値と=という文字とCの値をこの順で印刷する。

④プログラムおわり。

⑤今のプログラムを実行せよ、という命令。

⑥、⑦AとBの値として何かの数値を入力せよとマイコンが要求してくる。

⑧今のプログラムを実行した結果。ここまででひとつの仕事のおわり。

⑨次に何か命令を入れるようにマイコン側からプロンプトを出している。

図2－12　ベーシックのプログラム例

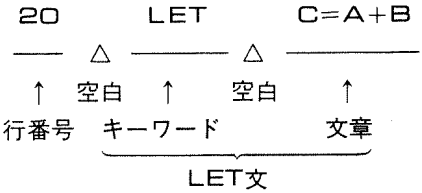


図2－13　文の形式

ません。しかし、この例で、ベーシックのプログラムとはどんなものか、そのイメージをつかんでください。

この例でプログラムにあたるのは、①～④までの4つの文です。それぞれの文は、行番号とキーワード、それから仕事の内容を指定する文からできています(図2―13参照)。ベーシックのプログラムは、基本的には、このような文章が組合わさってできたものです。

また、⑤行のRUNのようなものを、コマンドといいます。コマンドは、プログラムの中には書かれずに、プログラムの取り扱いについて、マイコンに指示を与えるものです。今のところは、RUN(プログラムを実行する)と、NEW(マイコンの中に書かれていた古いプログラムを消し、新しいプログラムの入力待ちの状態にする)と、LIST(作ったプログラムを画面に出力する)の3つのコマンドだけ覚えておいてください。

ベーシックの文

ベーシックのプログラムは、図2―13のような文の集合体ですが、その文としては、どのようなものがあるかを、例とともにみていきまし

よう。

ベーシックでは、それぞれの文は、必ず行番号から始っていなければなりません。

行番号

行番号とは文の頭にある4ケタ(9999)までの数字です。この数字のつけ方は任意ですが、実行されるときには、番号の小さい順（文を入力した順番ではなく）に実行されますから注意してください。したがって、図2—14の(a)と(b)は、全く同じプログラムとなります（(b)のプログラムをLISTコマンドで出力させると(a)のように並びかえられて出力される）。この機能は、ベーシックの特長ともいえます。そして、これがあるためにベーシックでは、プログラムの修正、改変がしやすくなっているのです。

```
(a) 10 INPUT A, B
    20 LET C=A+B
    30 PRINT C
    40 STOP

(b) 30 PRINT C
    10 INPUT A, B
    40 STOP
    20 LET C=A+B
```

図2—14　行番号と実行の順序

行番号は、順序が違っていなければ、どのようにつけてもかまいません。しかし、ふつうは、あとで文を挿入することができるように、10から始って、10おきに行番号をつけていくことが多いようです。

行番号の次には、キーワードを書きます。キーワードは、関数などを含め、40ほどあります（TK—80BS）。

次に、いくつかの基本的なキーワードの説明

をしましょう。ここでは、行番号を省略します。

ＰＲＩＮＴ

マイコンからの情報をＣＲＴに出力させるときに、このＰＲＩＮＴ文を使います。ＰＲＩＮＴ文の並びに変数があれば、その値、引用符（〃）でくくられた文字列があれば、その文字列を、そのまま出力します。

出力の仕方をコントロールする書式制御のためには、「；」、「，」、「ＴＡＢ」が使われます。その具体的な例については、図２－15を参照してください。細かい部分では、機種によって、書式制御の仕方が異なることもありますが、ほぼ、この例のようであると考えてよいでしょう。

書式制御に関する規則は次の通りです。

(1) 一行は、14文字づつのカラムに分けられています。

(2) 「；」は、次のカラムにカーソルを進めま

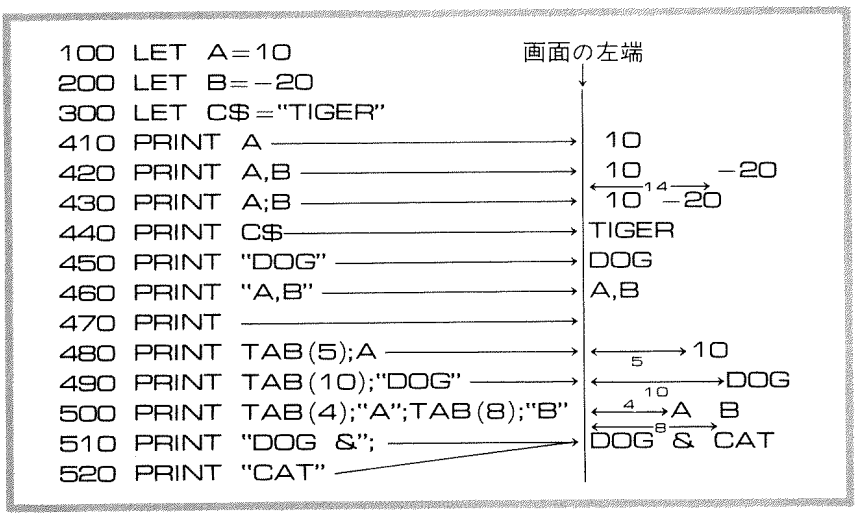


図２－15　PRINT文の使い方

ます。

(3)「;」は、カーソルを移動させません。

(4)「,」、「;」以外で、文が終っているときは自動的に改行します。

INPUT

PRINTの逆に、私たちがマイコンに何かを伝えたり、データを入力したりするときに、これを使います。プログラムの途中にこのINPUTがあると、実行はそこで一時中断されます。そしてマイコンは、マイコンを使っている人がそのINPUT文が要求しているもの（データなど）を入力してくれるのを待ちます。このとき、画面には「?」が描かれます。使用者がデータを入力すると、実行は再開されます。

```
① LET A=10
② LET B=20
③ LET C=A+B
④ LET A$="DOG"
```

図2－16　LET文の使い方

LET

LET文は代入文とも呼ばれます。LETとは、英語で「～させる」という意味を持っていて、ここでは、ある変数に、ある値を与えるときに使います。図2－16に示されているように、①、②では、A、Bという変数に、それぞれ10、20の値を与えます。したがって、このプログラムの中では、A、Bは、10、20として使うことができるわけです。③もやはり代入文で、これは、AとBの値を足して、それを変数Cの値にすることを意味しています。

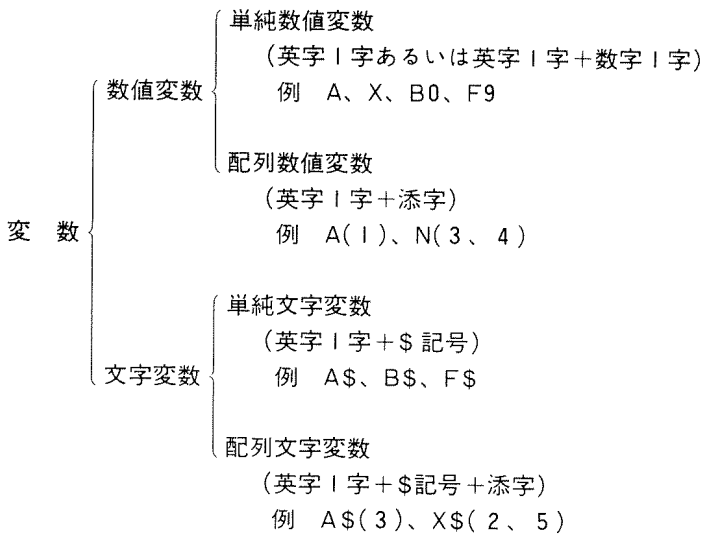


図２－17　変数の使い方

ここで、代入文に等号が使われていますが、これは、一般に数学で使われているものとは異なって、左辺と右辺が等しいことを意味していません。右辺の値を左辺の変数の値とせよ、という印であると考えてください。

変数

ここで、変数とは何かについて、説明しておきましょう。ベーシックの変数には、数値変数と文字変数の２つの種類があります。そして、それぞれに、単純変数としてと、配列変数としての２つの使い方があります。つまり、全部で４通りの使い方があることになります。

変数には、プログラムの実行中に、ある特定の値が与えられます。変数に値を与える文が、代入文（ＬＥＴ文）です。図２―17のように、文字変数に値を与えるには、その文字の両端を引用符でくくる必要があります。文字変数に代

入可能な文字データの長さには制限があるのが普通です。

IF—THEN

「もし～なら～せよ」という意味のキーワードです。条件に応じて、プログラムの流れを変える「判断」を行います。図2—18(a)の①は、もしAが1と等しかったら(Aが1だったら)、「GOOD」と印刷せよ、Aが1でなかったら、次の文に進め、ということを意味しています。

②はBが0でなかったら、行番号500の文に進み、Bが0だったら、次の文に進め、ということを意味しています。

一般に、IF—THEN文は、図2—18(b)に示されているように、もし条件が成立したら、THEN以下を実行(THENの後の行番号とおなじ文を実行するかTHENの後におかれた行番号に進む)し、条件が成立しなければ、THEN以下を無視して、次の行番号の文を行うことを指示する文です。ここで、条件にあたるものとしては、図2—18(c)に示されるような6種類の判断記号があります。

(a) ① IF A=1 THEN PRINT "GOOD"
② IF B<>0 THEN 500

(b) IF—THEN文の基本型

IF条件 THEN { 行番号なしの文 / 行番号

(c) 条件判断

A<B	AがBより小さいとき成立
A>B	AがBより大きいとき成立
A=B	AとBが等しいとき成立
A=<B	AとBが等しいかAがBより小さいとき成立
A=>B	AとBが等しいかAがBより大きいとき成立
A<>B	AとBが等しくないとき成立

図2—18 IF—THEN文の使い方

```
10 INPUT A
20 IFA=1 PRINT "グー"
30 IFA=2 PRINT "チョキ"
40 IFA=3 PRINT "パー"
50 STOP
```

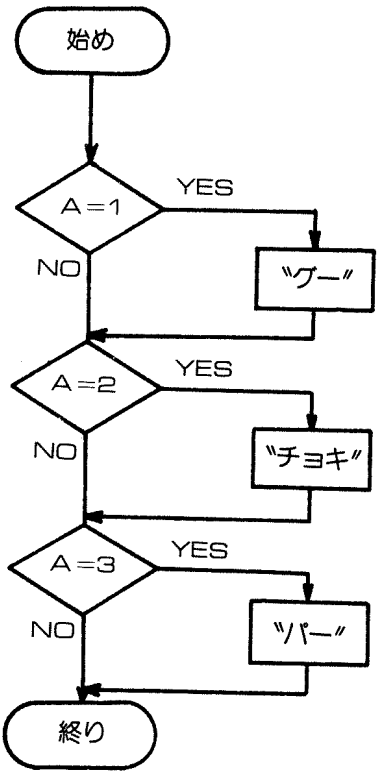


図2－19　IF－THEN文の実例

IF—THEN文は、図2—19のように、使われます。このプログラムを実行し、Aの値として、2を入力すると、20番の文では条件「A＝1」が成立せず、「グー」は出力されません。30番の文では、条件「A＝2」が、成立しますから、「チョキ」という文字が出力されます。40番の文でも条件は成立しません。50番の文にたどりつくと、プログラムは、おしまいです。

このプログラムで、Aの値として1、2、3以外の数を入力すると、文番号20、30、40の文すべてで、条件が成立しませんから、何も出力しないまま、実行が終了してしまいます。

GOTO

このGOTO文もIF—THEN文と同じよ

うに、プログラムの流れを変える役目を持っています。しかし、IF—THEN文では、その条件が成立するかしないかで、行先が変りましたが、GOTO文では、無条件に流れを変えます。

GOTO文の使い方には、図2—20の(a)、(b)に示されるように二通りのものがあります。(a)では、行番号110の文から、行番号140の文まで飛びます。つまり、行番号100の文の次に、140の文が実行され、120、130の文は、このときには実行されません。(b)は、あることを何回もくり返し行うようなときに使われます。行番号140のGOTO文によって、ふたたび行番号110の文に戻るわけです。したがって実行は、行番号100、110、120、130、110、120、130、110、120、130、110……の順に行われます。このようなくり返しをループ(輪)といい、110～130までの文で、ループを抜け出さない限り、くり返しを続けます。

GOTO文は、プログラムの流れを変えるために必要なものですが、あまり安易に使うと、プログラムがわかりにくくなります(特に(a)のような使い方)。プログラムを作るときには、フローチャートを書いて、その構造をできるかぎり簡単にするように心がけてください。

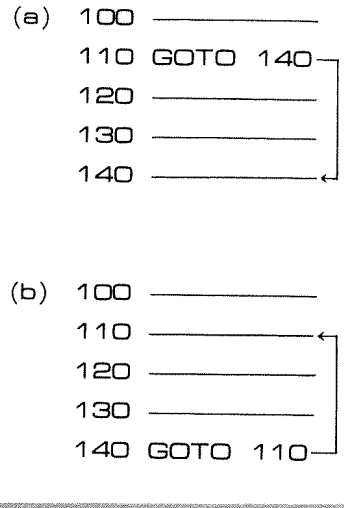


図2－20 GOTO文の役割

(a) ON 式 GOTO

行番号、行番号、……

(b) 90 ON A+1 GOTO
100,110,120
100 文①
110 文②
120 文③

図2－21 ON－GOTO文の使い方

ON－GOTO

これもGOTO文の仲間ですが、図2－21(a)のような形をしていて、ONのあとの式の値によって行先が変ります。具体的には、図2－21(b)のように使われ、式A＋1の値を4捨5入して整数にしたとき、それが1ならば行番号100に飛び、文①以下が行われます。また、2ならば、110に飛び、3ならば、120に飛びます。Aの値が0以下か、行番号の並びの数（この例では3）より大きくなると、エラーになります。

GOSUB

プログラムの途中で、同じようなことをくり返し何度もやることが必要になることがあります。そのようなときには、それをサブルーチンとして1つにまとめておくことができます。

たとえば、図2－22(a)のように、110～130行と160～180行で、全く同じ作業Aをしなければならないときには、それをまとめてサブルーチンとしてプログラムの最後の方に置き、図2－22(b)のように110行、160行にGOSUB文を使えばよいのです。ただし、サブルーチンの最後には、必ずRETURN文を置くことが必要です。

GOSUB文はGOTO文に似ていますが、

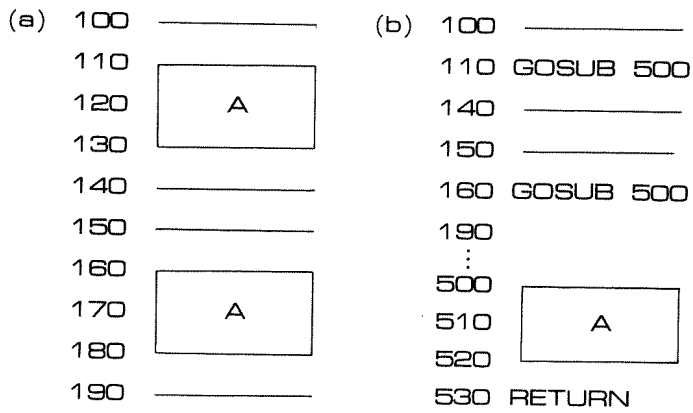


図2－22　サブルーチンの使い方

その違いを具体的に示すと次のようになります。図2―23(a)では、10～30行まで3つのGOTO文がありますが、最初のGOTO40によって、行番号40に飛んでしまい、ABCという文字を1回出力しただけでストップしてしまいます。しかし(b)のようにGOSUB文を使えば、50行でABCと出力したあと、そのGOSUB文の次の行に戻るので、結局、ABCという文字が3回出力されることになるのです。

```
(a)  10 GOTO 40
     20 GOTO 40
     30 GOTO 40
     40 PRINT "ABC"
     50 STOP
     実行の順序  10→40→50

(b)  10 GOSUB 50
     20 GOSUB 50
     30 GOSUB 50
     40 STOP
     50 PRINT "ABC"
     60 RETURN
     実行の順序  10→50→60→
     20→50→60→30→50→
     60→40
```

図2－23　GOSUB文GOTO文

FOR－NEXT

くり返し同じことをするために、GOTO文が使えるということはすでに説明した通りですが、FOR－NEXT文を使えば、GOTO文を使うより簡単に、くり返しのプログラムを作ることができます。

このFOR－NEXT文は、いままでにでてきたキーワードの中で、もっとも複雑な形をしています。一般形は図2－24に示された通りで、FOR文とNEXT文が対応して1組となっています。その意味は、FOR文からNEXT文までの間を、FOR文の中の変数（これを制御変数といいます）を、初期値から始めて増分（1回ループを回るごとに変化させる値）づつ増やして（減らして）行って、終値になるまでくり返す、ということです。

具体的には、図2－25のように使われます。

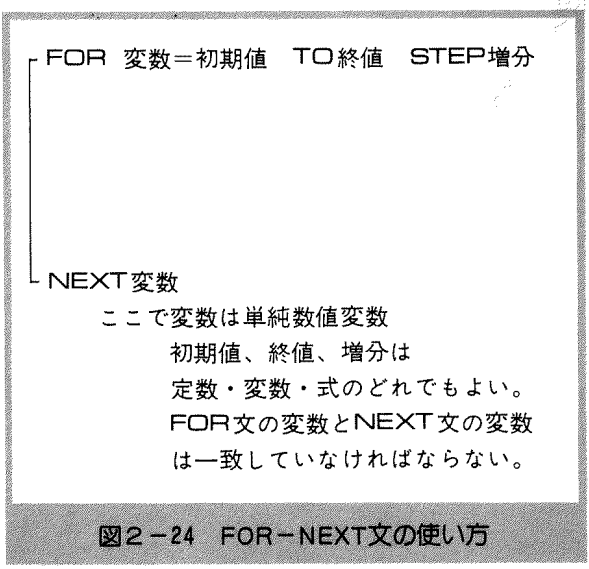


図2－24 FOR－NEXT文の使い方

①では㋐の部分を1回実行するごとに、制御変数Aの値が、初期値1から増分1づつふえて行き、Aの値が10になったとき（㋐を10回くり返したとき）このループを抜け出します。同様に、②のように増分が負の値のときには、㋑を1回

実行するごとに、制御変数Bの値が1づつ減って行きます。③のように実数を使うことも、④のように計算式を使うこともできます。またここで、①のように増分が1のときは、STEP以下を省略することができます。

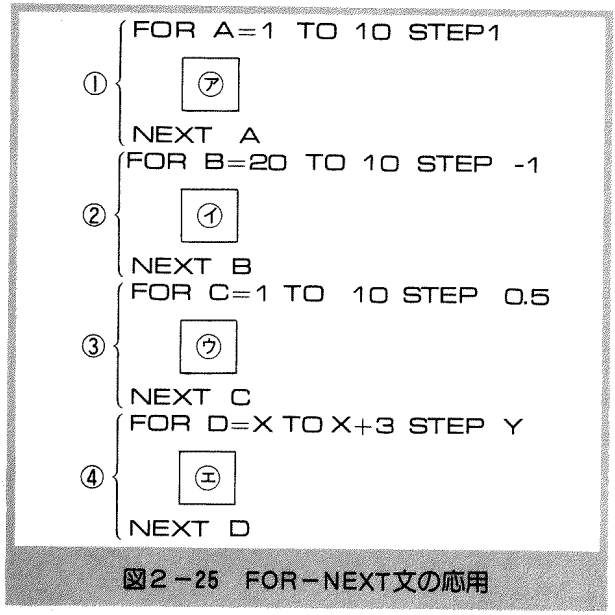


図2－25　FOR－NEXT文の応用

FOR—NEXT文を使うときに注意すべきことは、64頁・図2—26にまとめておきました。FOR—NEXTのループの中に、もう1つのループを入れることはできますが、そのときには、図2—26の(a)のような、入れ子（ネスト）といわれる構造を取っていることが必要です。

また、(c)のようにループから飛び出すことはできますが、(d)のようにループの中に飛び込むことはできません。(c)でループから飛び出したときには、制御変数は、そのときの値を保っています。

また、FOR文の初期値、終値、増分の間に矛盾があってはなりません。

ループの中で制御変数を使うことはできますが、(f)のように、その値を変えてはいけません。単純数値変数ならば制御変数として使えますが、このような間違いをおこさないように、制御変

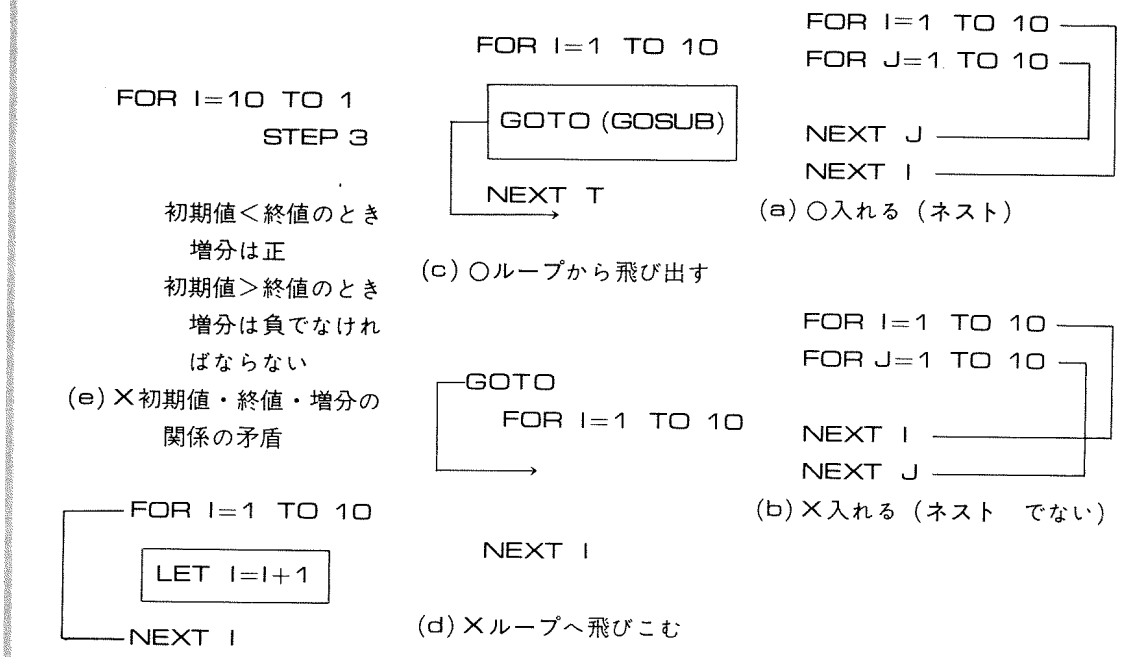


図2－26　FOR-NEXT文を使うときの注意

数として使う変数を定めておくとよいでしょう。一般には制御変数としては、I、J、Kなどがよく使われているようです。

READ・DATA・RESTORE

READ文もINPUT文と同じように、変数に数値（文字）を与えるための文です。しかしINPUT文が外部（たとえばキーボード）から、データを入力するための文であったのに対して、READ文は、同じプログラム中のDATA文から、データを受け取ります。DATA文は、プログラム中のどこに置いてもよく、1つのプログラムのDATA文のデータは、1つのデータの列を作り、前から順にREAD文中の変数に代入されて行きます。

図2—26のプログラムでは、10行のREAD文によって、100行のDATA文からA、B、Cに、それぞれ10、20、30という値が代入されます。

そして、190行のREAD文によっては、100行と120行のDATA文からD、E、Fに、それぞれ40、50、60という値が代入されます。

RESTORE文は、READ文によって読み込まれるべきデータの値を、そのプログラムのDATA文の最初のデータに戻す働きをします。したがって図2—27のプログラムの190行の前にRESTORE文を挿入すると（図2—28）、

```
10 READ A,B,C
    ⋮
100 DATA 10,20,30,40
    ⋮
120 DATA 50,60
    ⋮
190 READ D,E,F
```

図2—27　DATA文とREAD文

```
10 READ A,B,C
100 DATA 10,20,30,40
120 DATA 50,60
180 RESTORE
190 READ D,E,F
```

図2－28　RESTORE文

190行のREAD文は、100行のDATA文からデータを読み込み始め、結局、D、E、Fには、それぞれ10、20、30という値が代入されます。

REM

REMは、REMARKの略で、注釈を意味しています。REMで始まる文は、マイコンは読み飛ばしてしまい実行はしません。しかし、プログラムのリストには残りますから、プログラムの題名や説明を書く場合に利用できます。

あまり安易にREMを使うことは、マイコンの記憶量が少ないことなどを考えると、望ましいことではありませんが、昔書いたプログラムや他人の書いたプログラムは、なかなかその意味が汲み取りにくく、みにくいものです。適度にREMを使い、読みやすく、わかりやすいプログラムを作りたいものです。

DIM

配列変数を使用する前に、その大きさを宣言するための文が、このDIMです。配列変数とはA(n)、B(m, n)の形で表わされる1次元、あるいは2次元のデータのまとまりです。どのデータを使うかは括弧の中（添字）の値を何にするかによって決まります。添字の下限は、0です(下限が1のシステムもあります)。したがって図2—29の場合、1次元の配列Aは、A(0)

からA(10)までの11個の数値を持ち、2次元の配列Bは、B(0，0)からB(3，5)までの24個の数値を持つことができます。数値変数の配列だけでなく、文字変数の配列も使えます。

```
10 DIM A(10),B(3,5)
20 DIM C$(10)
```

図2－29　DIM文の使い方

組込み関数

ベーシックには、いくつかの関数が、組込み関数として用意されています。ここでは、乱数発生のためのRNDと、数値を整数にするINT、平方根を求めるSQRについて説明します。

(1)　RND　ゲームなどのプログラムでは、乱数を必要とすることがあります。このようなときに、RNDという関数が利用されます。RNDは引数を持ちません。その値は、0以上1未満の一様乱数です。RNDは、図2—30のような形で利用されます。

```
LET A=RND
PRINT RND
```

図2－30　関数RNDの使い方

RNDを利用するときに注意することは、乱数発生のために使われている方法から、図2—31(a)のプログラムを走らせてみると、そこから生じる乱数の系列が常に同じになってしまうことです。

```
(a)  10 PRINT RND
     20 GOTO 10

(b)  10 RANDOMIZE
     20 PRINT RND
     30 GOTO 20
```

図2－31　RANDOMIZE文

```
INT (1.5) =1
INT (−0.3) =−1
INT (16.0) =16
INT (−7) =−7
```

図2－32　関数INTの使い方

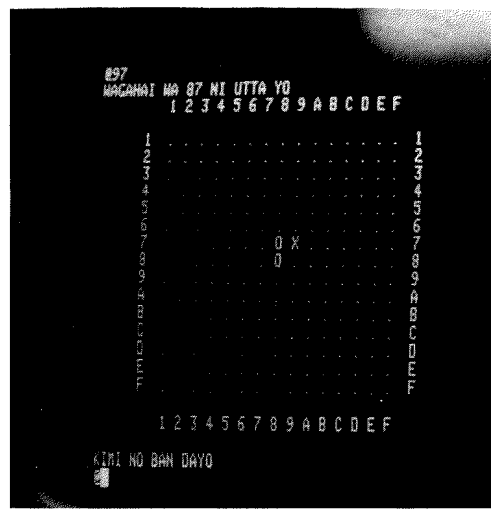
それを避けるために、ゲームなどで乱数を使うときには、図2―31(b)のようにRANDOMIZE文を、文頭に挿入することが必要です。

(2)　INT　INTは1つの引数Xを持ち、値は、Xをこえない最大の整数となります。つまり、図2―32のようになります。

(3)　SQR　SQRは、引数を1つ持ち、値は、その平方根になります。このとき引数が、負の数だとエラーになります。

以上が、ベーシックの主なキーワードや関数の概要です。ほかにも重要な機能はありますが、これだけでも十分プログラムは書けますし、また読むこともできるはずです。

以下の章のプログラムで、ここに記さなかったような機能を使うときには、そのつど説明することにします。



マイコンの画面

マイコンの働き

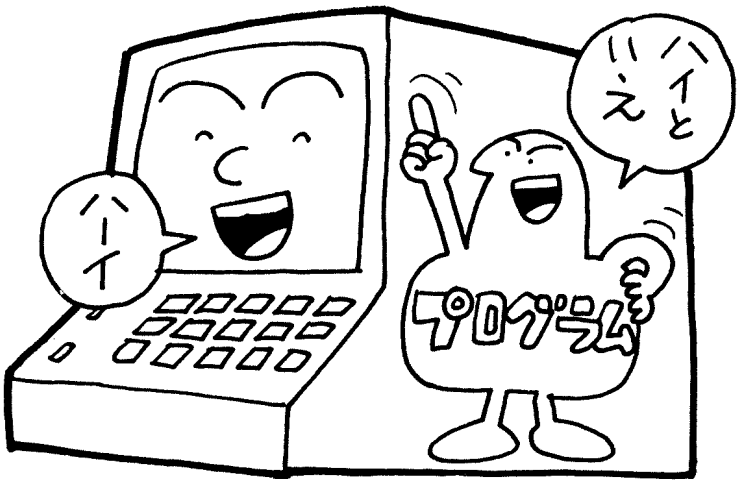
マイコンの本体、つまり周辺機器を取り除いたマイコン自体どんな働きを持つのでしょうか。マイコン本体の基本的な働きについては知っておく必要がありますので、ここで簡単に説明しておきましょう。

ベーシック・インタプリタ

周辺機器を取りつけないマイコン、つまりマイコン本体は、それ自体、複雑な働きをします。キーボードからの入力を受けつけ、CRTの表示をするのは、すべて、このマイコン本体が行います。

さらに、このマイコン本体の中でも、もっとも重要な働きをするのがCPU（中央処理装置）

マイコンの中のマイコン(CPU)



プログラムはマイコンに住みついて命令する力をもつ

と呼ばれるもので、このCPUは「マイコンの中のマイコン」と呼ぶことができます。

したがって、マイコン本体は、CPUを除いた部分とCPUの2つに区分することが可能です。

仮にCPUを除いた部分を第1のマイコン、CPUを第2のマイコンと呼ぶことにすると、私たちが使うのは第1のマイコンとなります。

つまり、私たちは、マイコンに仕事をさせるために、プログラムを書き、それをキーボードから入力します。すると、第1のマイコンから入ったプログラムは、次に第1のマイコン自体が第2のマイコンであるCPUに仕事を命令するのです。

プログラムは、入力された時点で、私たちの手から離れて、第1のマイコンに住みつき、CPUに対する「命令する力」をもつといってよ

いでしょう。

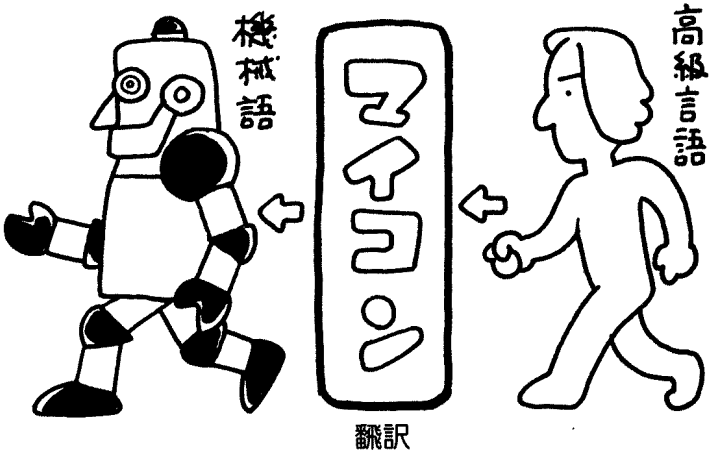
ところがCPUは、機械語という特殊な言葉しか理解することができません。

私たちが、キーボードから入力するプログラムは、普通、ベーシックのような高級言語で書かれていますから、プログラムは、入力されたそのままの姿では、CPUに命令を与えることができないのです。

そこで、マイコンは、ベーシックのような高級言語で書かれたプログラムを、マイコンの中で機械語のプログラムに直します。高級言語のプログラムから機械語のプログラムへの翻訳がマイコンにとって必要になるわけです。

マイコンが行う、この「翻訳」も1つのプログラムの仕事なのです。このことをインタプリトとか、コンパイルと呼びます。

また、コンパイラやインタプリタが翻訳しよ



高級言語はマイコンの中で機械語に翻訳される

うとする高級言語のプログラムをソース・プログラム、翻訳した結果の機械語のプログラムを、オブジェクト・プログラムといいます。

プログラムと制御

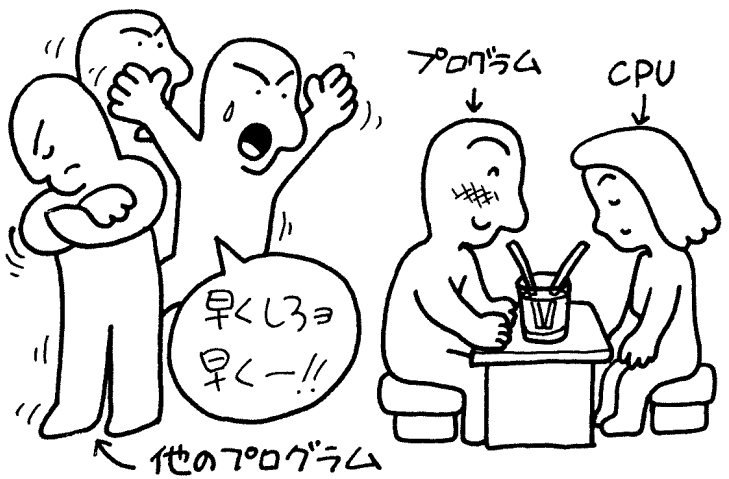
プログラムというのは、マイコンに、どのように仕事を進めていくかという詳しい指示です。機械語で書かれたプログラムなら、どんなプログラムでも直接CPUに仕事を命令することができます。コンパイラやインタプリタも機械語で書かれたプログラムです。そして、CPUに、ソースをオブジェクトに翻訳するという仕事を指示します。

機械語で書かれた1つのプログラムが「実行されているということは、そのプログラムがCPUに仕事をさせているということです。2つのプログラムが、同時にCPUに仕事を命令することはできません。いくつかのプログラムが、交代でCPUに仕事をさせるときでも、必ず「一度に1つ」のプログラムがCPUを使っています。プログラムが実行されているとき、つまり、プログラムがCPUを自分の仕事のために使っているとき、そのプログラムに「制御がおかれている」ともいいます。

コンパイラとインタプリタ

翻訳作業の性格と、翻訳、実行時の「制御」の移り方が、コンパイラとインタプリタの違う点です。

コンパイラは、翻訳をしている間だけ制御を確保しています。作業が終ると、制御を捨ててしまい、そのあと制御が、どのプログラムに移るかは、コンパイラの知ったことではありません。コンパイラの翻訳したオブジェクトは、もう、一人だちで仕事ができる立派な機械語のプログラムですから、心配する必要もないわけで



1度に1つだけのプログラムがCPUを使う

す。コンパイル終了後、すぐにオブジェクトに制御が移る形式のシステムを、「コンパイル・アンド・ゴー」のシステムといいます。

インタプリタは、最初から、最後まで、制御を自分のものにしておきます。ソース・プログラムを、機械語のオブジェクトとして一人だちさせるのではなく、ソースにしたがって、自分がCPUに仕事を命令してしまうのです（74頁・図2—33）。

ベーシック・システムは、インタプリタ・システムです。インタプリタ・システムの利点は、プログラムの作成から実行までの過程が、すべて1つのプログラム（インタプリタ）の制御下にあって、私たちが「制御」の置き場所に頭を使う必要がないことです。

コンパイラ形式のプログラム言語を使うときは、ソース・プログラムを作り、コンパイルし、

実行するという手順のそれぞれを異ったプログラムの制御下で行わなければなりません。私たちは、そのたびに、制御を渡そうとするプログラムを指定する（そのプログラムを「呼び出す」と表現します）わけです。この手順は、意外にわずらわしく、特に、呼び出すプログラムが、カセット・テープのような外部記憶媒体に保存されるシステムでは、装置の操作にも、かなり

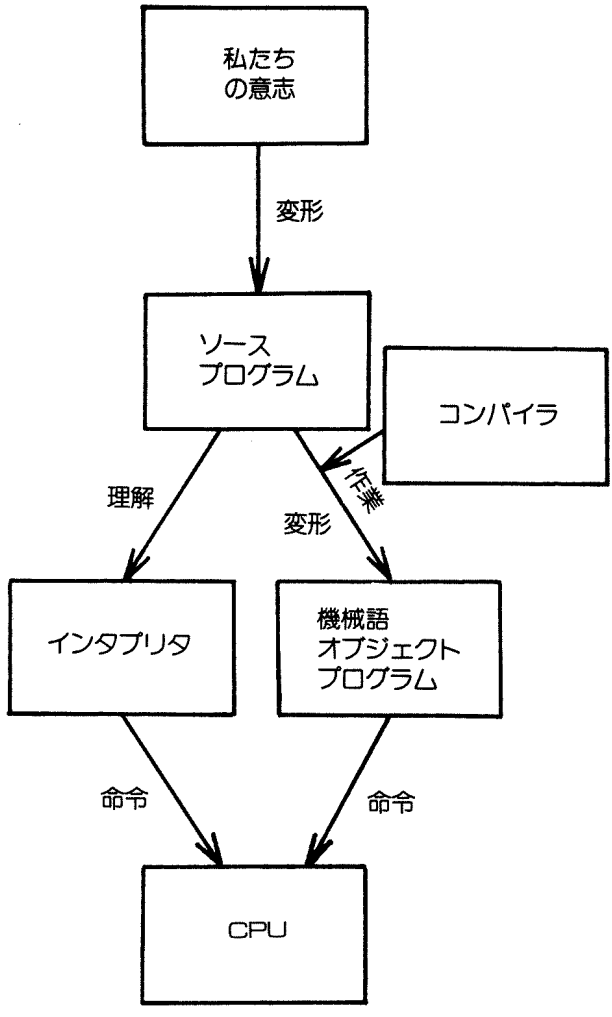


図2－33　インタプリタとコンパイラの違い

の手間がかかります。

ベーシック・インタプリタは、私たちが、マイコンを使っている間、厄介な操作をすべて引受けてくれます。「使いやすさ」では、インタプリタ・システムの方が、はるかにすぐれているといえるでしょう。

モニタとユーティリティ

モニタも、ユーティリティも、機械語で書かれたプログラムで、システムに組み込まれています。ユーティリティは、私たちがシステムを使いこなすために必要な、いくつかのプログラムの集まりです。テキスト・エディタ、ファイル・マネジメントがユーティリティの例です。

モニタ　マイコンに電源スイッチを入れた瞬間からCPUは活動を始めます。このとき、CPUを制御下におくプログラムがどこにもなければ、CPUは何を始めるかわかりません。CPUを、一番最初に押え込んでしまうのが、モニタです。私たちがベーシックを使いたいときは、このモニタに「ベーシックを使いますよ」と通知してやるのです。モニタは、素直に、ベーシック・インタプリタに制御を渡してくれます。ベーシックを使うのをやめて、ほかの高級言語を使いたいときや、ユーティリティを使いたいときにも、モニタに通知します。プログラムに重大なエラーが起ったり、私たちが「BREAK」キーを押したときにも、やはり制御がモニタに移ります。ですから、「BREAK」キーは、現在実行中のプログラムを、一時中断したいときに用いることができるわけです。モニタは、システム全体の交通整理を担当しているといえるでしょう。

テキスト・エディタ　私たちが、アセンブリ

言語やコンパイラ言語でプログラムを作るときには、ベーシックのときのように、簡単に修正することができません。アセンブラやコンパイラ（両方とも似たようなものです）は、プログラムが全部できてからでなければ、翻訳を受けつけませんし、全部をみた上でエラーを指摘してくるからです。プログラムを作ったり、修正したりするのに便宜をはかってくれるのが、テキスト・エディタです。

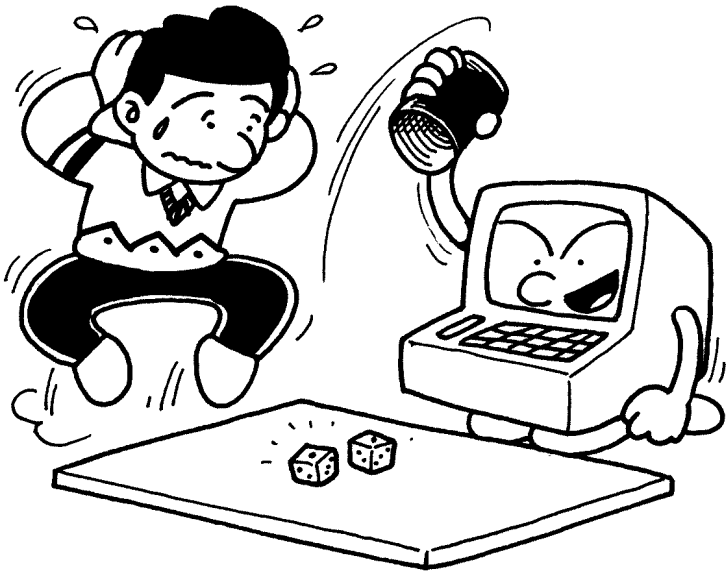
ファイル・マネジメント　オーディオ・カセットや、フロッピーディスクに入れたプログラムやデータは、直接目にみえる形になっていません。どんなデータがどのように入っているのかを調べること、そして現在マイコンが持っているデータを、どんな形でこれらの外部記憶媒体に入れるのかを決めること、それが、ファイル・マネジメントの仕事です。ファイル・マネジメントは、データ・ベースという、大量のデータを扱う技術と深い関係があります。現在のマイコンでは、まだ能力不足かもしれませんが、これからに期待することができそうです。

以上で説明したモニタ、各種ユーティリティそれにベーシック・インタプリタのような言語プロセッサなどを総称して、システム・プログラムといいます。マイコンは、これらのシステム・プログラムに支えられて、その能力を発揮することができるのです。

BASIC FOR MICROCOMPUTERS

第 3 章

ゲームのプログラムと遊び方



サイコロ・ゲーム

サイコロには、1から6までの目が刻まれています。サイコロをころがすと、1から6までのどれかの目が出るというわけです。

サイコロの面白さは、そのデタラメさにあります。サイコロを振って、6つの目のうちどれが出るかは、まったくわかりません。1が出る可能性も、6が出る可能性も、同じだけあって、特に3の目が出やすいなどということもありません。「どの目が出るかは、まったくわからない」という意味で、サイコロは、「デタラメ」なのです。

「デタラメ」を応用したサイコロ・ゲームには、さまざまな種類があります。

このサイコロ・ゲームを、マイコンを使って

楽しんでみましょう。

マイコンの中でデタラメさを作り出すのは、「RND」という関数です。

RNDの使い方

関数RNDは、マイコンの機種によって使い方が違いますが、ここでは、

RND(1)

によって、新しい乱数をつくることにします。乱数は、0と1の間の数です。変数Xに、乱数を代入したければ、

LET X=RND(1)

という文を、実行します。図3—1のプログラムを実行すると、図3—2のような出力が得られます。

このRNDは、本当の「デタラメな数」を作っているのではありません。システムによって違いはありますが、デタラメでない方法で作り、私たちには、デタラメのようにみえるだけです。それでも、ゲーム用の「デタラメ」を作り出すには十分です。まず、サイコロを、このRNDで作ってみましょう。

```
100 REM RANDOM
110 LET X=RND(1)
120 PRINT X
130 GOTO 110
140 END
```

図3－1　RNDを使ったプログラム

```
・238976
・842751
・223466
・488254
・372966
・584237
・932249
・134561
・372248
```

図3－2　図3－1のプログラムの実行例

マイコン・サイコロ

サイコロは、ただ、ころがしているだけでも楽しいものです。サイコロが、いくつかあれば、その目の出方のパターンもさまざまですし、全部の目が1だったりすると、何となく珍らしいことが起こったように感じます。

ところで、ここではマイコンにサイコロを振ってもらいましょう。

```
100 REM サイコロ
110 PRINT "サイコロ"
120 PRINT
130 FOR I=1 TO 6
140 L=INT (RND (1) *6) +1
150 PRINT L;
160 NEXT I
170 PRINT
180 GOTO 130
999 END
```

図3－3　サイコロのプログラム

プログラムは簡単で、図3―3のようになります。1行に、6個のサイコロの目を出力していきます。6個のサイコロの目は、いろいろなパターンを作ります。同じ目が3つずつとか、2つずつになることもあるでしょう。全部同じ目というのは、どういうわけか、出にくいようです。目の数が、2、3、4のように続いた組が2組とか、1、2、3・4、5、6が、1つずつになるかもしれません（図3―4）。

丁半ゲーム

サイコロを2個出して、丁半ゲームのプログラムを作るのも簡単です。持ち点を1000点にして、点を賭けるようにしてみましょう。丁（偶数）

```
サイコロ
 4   2   3   6   2   1
 3   6   5   2   1   3
 4   5   2
```

図3－4　図3－3のプログラム実行例

```
100 REM ODD OR EVEN
110 LET S=1000
120 PRINT "モチテン ";S
130 PRINT "カケテン ハ"
140 INPUT B
150 PRINT "チョウ カ ハン カ"
160 INPUT L
170 LET X=INT (6*RND (1)) +1
180 LET Y=INT (6*RND (1)) +1
190 LET E=X+Y-2*INT ((X+Y)/2)
200 PRINT
210 PRINT X;Y;
220 IF E=1 THEN 250
230 PRINT "  ノ チョウ"
240 GOTO 260
250 PRINT "  ノ ハン"
260 PRINT
270 IF L=E THEN 300
280 LET S=S-B
290 GOTO 120
300 LET S=S+INT (0.8*B)
310 GOTO 120
320 END
```

図3－5　丁半ゲームのプログラム

```
モチテン　　1000
カケテン　ハ
?100
チョウ　カ　ハン　カ
?1

　４　　６　　ノ　チョウ

モチテン　　900
カケテン　ハ
?50
チョウ　カ　ハン　カ
?1
　２　　４　　ノ　チョウ
```

図３－６　丁半ゲームの遊び方

に賭けるときは「0」、半（奇数）に賭けるときは「1」を入力します。勝つと、賭けた点の8割がもらえます。

プログラムは（81頁・図3―5）、出力の様子は図3―6に示します。

このプログラムには、少し説明が必要かもしれません。81頁の図3―5の190番の文は、出た目（XとY）の和が、偶数か、奇数かを判定するものです。X＋Yが、たとえば、6のときは、

(X＋Y)/2＝3

INT((X＋Y)/2)＝3

2＊INT((X＋Y)/2)＝6

X＋Y－2＊INT((X＋Y)/2)＝6－6＝0

に、なります。X＋Yが9なら、

(X＋Y)/2＝4.5

INT((X＋Y)/2)＝4

2＊INT((X＋Y)/2)＝8

X＋Y－2＊INT((X＋Y)/2)＝9－8＝1

というわけです。

結局、X＋Yが偶数（丁）のときは、E＝0、奇数（半）のときは、E＝1になり、このEと、私たちの入力L（160で入力されている）を比較

して（270）勝負を決めます。

持ち点は、110番の文で、最初1000にされ、280と300で、精算されます。300で、

INT(0.8＊B)

としたのは、「賭け点Bの8割を支払う」という規則があるからです。1点しか、賭けないと、

INT(0.8＊1)＝INT(0.8)＝0

ですから、勝っても、持ち点は増えません。

「チョウ　カ　ハン　カ」と聞いてきたときに、1と0以外のものを入力すると、必ず負けてしまいます。

270で、L＝Eが、成立せず、そのときは、280は、「負け」の場合の持ち点精算をしているのです。

「丁半」を、何人かで遊ぶようにプログラムを修正することもできるでしょう。持ち点精算が、いくらか面倒になるだけです。

サイコロを使ったゲームには、もう少し複雑なものもあります。

そのプログラムを紹介する前に、CRT画面のコントロールについて、すこし説明しておきましょう。

画面の更新

CRTの画面に表示できる文字、記号の数には限りがあります。1行単位で、入出力を行うベーシックでは、16行がその限度です。16行を使いきって、画面の上から下まで文字でいっぱいになったとき、次の出力は、どこに表示されるのでしょう。

いくつかの方法を、考えることができます。

① 画面全体を少し上の方に移動させて、下にスペースを作る。画面の最上部は消えてしまうが、大部分は残っているし、新しく表示さ

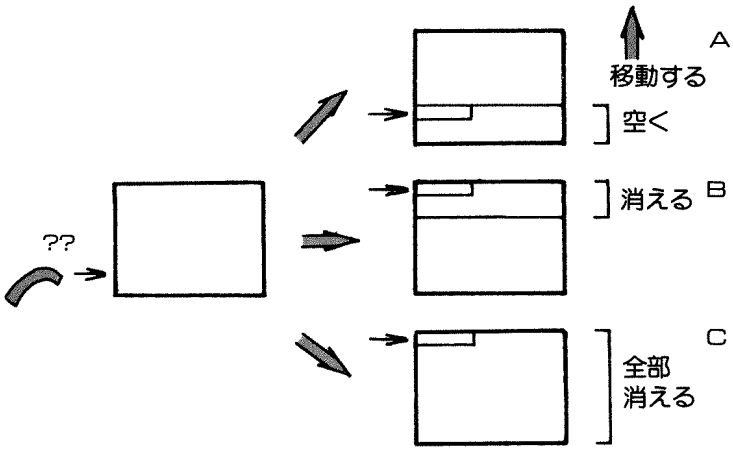


図3－7　画面の更新

れるものが、以前に表示されていたものとつながるのでみやすい（図3―7A）。

② 画面の上の方を何行か消して、スペースを作る。画面は移動させない（図3―7B）。

③ 画面全部を一度消してしまう。新しく最初の行の先頭から表示していく（図3―7C）。

CLEAR文

マイコンでは②と③が多く、これは図表などの表示には不便な方法です。図表そのものは画面に収まるようにしても、書き始める位置によっては消されてしまったり、途中で分けられてしまったりするからです。その点を考えて、②と③の方式を使うマイコンには画面全部を消すと同時に、カーソルを最初の行の先頭（つまり、画面の一番左上）に持ってくる命令が用意されています。図表などを書くときには、まずこの

命令を実行すれば最初の行から書き始めることができます。画面を有効に使うことができるわけです。

命令の形は、機種によって違いますが、

CLEAR（クリア）
EJECT（イジェクト）

のように呼ばれていることが多いようです。EJECT文は、ラインプリンタのページがえにもよく使われます。ここでは、CLEAR文によって、画面を新しくする（クリアする）ものとします。

例えば図3—8のプログラムを実行すると、画面は、86頁の図3—9～11のように変っていきます。

「跳び跳び」運動

ゲームのプログラムでは、画面に図形を描くことが多くなります。図形を「動かす」ことも必要になるでしょう。ところが、CRTの画面というのは16×32（16×64）の区画に分けられていて、文字・記号は、その区画の中に、きち

```
 10 REM CLEAR
 20 CLEAR
 30 FOR I=1 TO 9
 40 PRINT TAB (I) ;I
 50 NEXT I
 60 CLEAR
 70 FOR I=9 TO 1 STEP -1
 80 PRINT TAB (I) ;I
 90 NEXT I
100 STOP
110 END
```

図3－8　カーソル制御のプログラム

```
>80 PRINT TAB (I) ;I
>90 NEXT I
>100 STOP
>110 END
>RUN
```

図3－9　図3－8のプログラム実行例1

```
 1
  2
   3
    4
     5
      6
       7
        8
         9
```

図3－10　図3－8のプログラム実行例2

```
         9
        8
       7
      6
     5
    4
   3
  2
 1
```

図3－11　図3－8のプログラム実行例3

んと収まっていなければなりません。連続的に位置を変えることができないのです。

文字・記号が「動いた」ように見せかけるためには、その区画に表示されていた文字・記号を消して、その直後に、隣の区画に同じ文字・記号を表示します。文字・記号を「跳び跳び」に出していくわけです。この「跳び跳び」運動は、表示のタイミングを慎重に選べば、実際の連続的な運動と全く同じようにみえます。この現象を、「仮現運動」といって、学問的にも詳しく研究されています（図3―12）。

もっとも、表示のタイミングを細かくコントロールするためには機械語でプログラムを書かなければなりません。ベーシックのプログラム

で、そこまでコントロールすることは難しいようです。

CURSOR文

ベーシックで、この「跳び跳び」運動のプログラムを書くには、

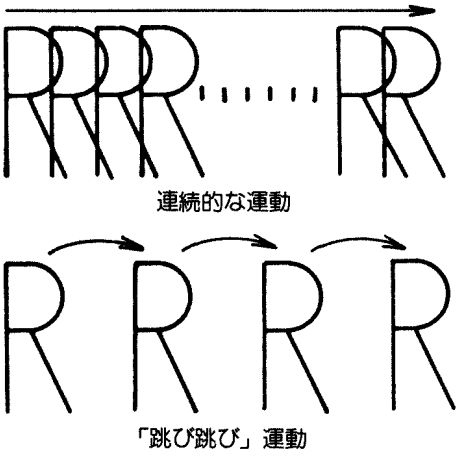


図3－12　画面上の運動

① 位置Xの文字を消す
② 同じ文字を位置Xに表示する

という手順を繰返せばよいのです。そのためには、画面上の「位置X」を指定する文が必要になります。多くのマイコンでは、

CURSOR（カーソル）

という文で、「位置」を指定します。

CURSOR A、B

という文を実行すると、カーソルが、画面の上からA行目、左からB字目に移動します。今までカーソルがあった位置とは無関係に、突然、「ジャンプ」するのです。カーソルは、「次の文字・記号はここに出るよ」という印でしたから、CURSOR文によって画面上の好きな位置に文字・記号が書けることになります。

たとえば、

CURSOR 12、27

PRINT "＊"

を実行すると、画面の上から12行目、左から27字目の位置に「＊」が表示されることがわかるでしょう。

今まで、ある位置にあった文字を消すためには、その位置にカーソルを移動させてから「空白(ブランク)」を書きます。たとえば、前の例で表示した「＊」を消すには、

CURSOR 12、27
PRINT " "

を実行するわけです。

CURSOR文を使えば画面の表示を自由にコントロールできますが、プログラムは、わかりにくくなってしまいます。GOTO文がプログラムの流れを勝手に変えるように、CURSOR文も、カーソルの自然な動きを強引に変えてしまうからです。

CURSOR文を使った「跳び跳び」運動のプログラムを図3—14に示します。「＊」が、渦巻のように動きまわって、あまりみつめていると目を回してしまうかもしれません。図3—13は、「＊」の軌跡を描いたものです。画面を、図3—13のようにするには、図3—14の220番の文を除きます。

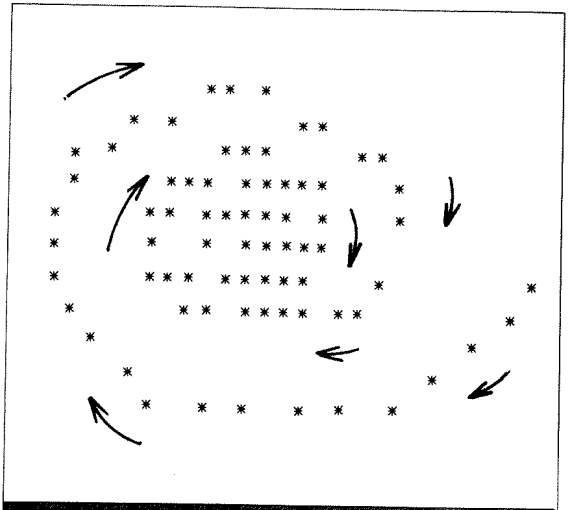


図3—13 "らせん"

```
100 REM SPIRAL
110 DIM X(2),Y(2)
120 LET N=1
130 LET T=0
140 LET X(1)=15
150 LET Y(1)=16
160 CLEAR
170 LET T=T+0.2
180 LET S=EXP(-0.11*T)
190 LET X(3-N)=INT(7*S*COS(T))+8
200 LET Y(3-N)=INT(15*S*SIN(T))+16
210 CURSOR X(N),Y(N)
220 PRINT " ";
230 LET N=3-N
240 CURSOR X(N),Y(N)
250 PRINT "*";
260 GOTO 170
270 END
```

図3－14 〝らせん〟のプログラム

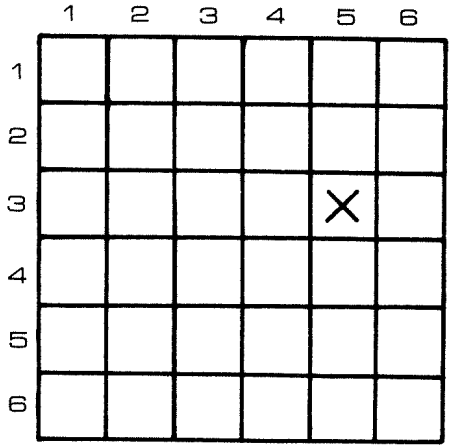


図3－15　点取りゲーム

点取りゲーム

サイコロを応用したゲームは、たくさんありますが、ここでは、2コのサイコロと、図3－15のようにタテ・ヨコが6つずつの区画に分れた表を使うゲームを取りあげました。

2コのサイコロを振って3と5なら、表の3行5列目（図3－15で「X」を書いたところ）にあるものが、サイコロを振った人のものになります。表のマス目には、前もって賞品を置いておいたり何か書き込んでおきます。サイコロを振って、「ミカン3個」と書いてあるマス目にあたれば、ミカンが3個もらえます。「歌を歌う」と書いてあるマス目にあたれば、歌を歌わなければなりません。ゲームとしては単純ですが、数人が参加して順番にサイコロを振っていくと、

なかなか楽しいものです。

2人で遊ぶときにはマス目に得点を書き込んでおいて、サイコロを振るたびに持点に加えていきます。たとえば、サイコロを振って、「5」というマス目にあたれば持点に5点をたすようにするわけです。2人が交互にサイコロを振っていって、持点が、先に目標点に達したほうが勝ちです。このゲームを、マイコンで楽しんでみましょう。

ゲームの楽しみ方

得点表

得点表は、−5から7までの数が6行6列（36個）に並んだものです。この表は、ゲームの最初に作られます。そして、たとえばサイコロ2個の目が4、5のときは、この表の4行5列目の数が得点になります。図3−16では「5」というわけです。−5から7までとしたのは、平均して得点が＋になるようにしたかったからです。得点表が一の得点ばかりだと、いつまでたっても得点が増えないことになります。もちろん、平均値が＋の側にあれば、−3から9まで、というように適当に範囲を決めてかまいません。

2つのサイコロ

サイコロの目が1から6まで、とい

4	−2	1	−3	1	−2
7	0	−1	7	−3	4
−1	3	3	−5	−3	0
2	−5	6	4	5	0
2	3	0	2	2	6
−3	4	−2	4	0	5

図3−16　得点表

のは当然ですが、マイコンのほうで勝手に振って私たちの得点を決めてしまうのでは面白くありません。そこで、マイコンの振るサイコロの数を4つにします。私たちが、この4つのうち2つを前もって指定すると、マイコンはサイコロの目を4つ出力して、私たちの指定した2つのサイコロで得点表のマスを選ぶのです。

マイコンは、4つのサイコロの目を、

4325

のように、4桁の数で出力するとします。そうすると、私たちがサイコロ2つを指定するときにも、

1010

のように、4桁の数を使う方がわかりやすいでしょう。「1」は、その位置にあるサイコロを選ぶという意味で、「0」は、選ばないという意味です。1と0で、その位置の数を選ぶことを、「マスをかける」ともいいます。

1010

というマスを

4325

にかけると4と2が得られます。この4と2を使って得点表の4行2列目の得点（図3—16では、5）を得点にするわけです。マスは、4つのサイコロの目が決まる前に指定しますから、4つの目が出てから「0110にしておくべきだった」と悔んでも、後のまつりです。もちろん、マスを慎重に考えてもしかたがありません。

勝ちと負け

このゲームは、私たちとマイコンとが、勝負を争うものです。交互に、サイコロを振って得点を決め、最初から加えていった得点が先に30点になったほうが勝ちです。30点を限度にしな

```
テントリ　ケ゛ーム

サイコロ　ヲ　4コ　フリ　マス
ソノウチ　ノ　2コ　テ゛　トクテンヒョウ　ノ　マス
ヲ　エラヒ゛　マス
ト゛ノ　2コ　テ゛　エラフ゛　カ　ハ　マエモッテ
キメテ　クタ゛サイ
2ハ゛ン　メ　ト　4ハ゛ン　メ　ノ　サイコロ　ヲ
エラフ゛　トキ　ハ　〝?〃　ノ　アト　ニ
〝0101〃　ト　ニュウリョク　スレハ゛　ヨイ
ノ　テ゛ス
ホ゛ク　モ　オナシ゛　コト　ヲ　シ　マス
サキ　ニ　30　テン　ニ　ナッタ　ホウ　カ゛
カチ　テ゛ス
```

図3－17　点取りゲームの実行例1

ければならないという理由はもちろんありません。1000点くらいにしてもいいでしょう。

遊び方

プログラムを入力し終って、実行を始めると、まず、ゲームの説明が画面に出てきます（図3—17）。

次に、画面が、一度消えて、得点表が出てきます。

```
ヤマ　ノ　ンベ
```

と表示されますから、「?」のあとに、4個のサイコロのどれを選ぶか、というマスクを入力します。

そのすぐ下にサイコロの目が表示され、得点も出てきます。そして累計得点が1番下の行に表示されるはずです（図3—18）。

次は、マイコンの番です。

```
ボク　ノ　ンベ
```

```
 4  *-2   1   -3   1   -2
 7    0  -1    7  -3    4
-1    3   3   -5  -3    0
 2   -5   6    4   5    0
 2    3   0    2   2    6
-3    4  -2    4   0    5

          キミ ノ ハ゛ン
?1100
 1231    トクテン  -2

          キミ  -2     ホ゛ク   0
```

図3－18　点取りゲームの実行例2

という表示が出てから、マイコンも私たちと同じことをします。つまり、マスクを1、0の4桁で選び、その下にサイコロの目が出るわけです。プログラムは非情ですから、マイコン側に有利なようにサイコロの目を出したりはしません（図3―19～20）。

選ばれた2つのサイコロが、2、5で、得点表の2行5列目を選ぶときは、そこに「＊」が出ます。この「＊」は、1回ごとに、出たり、消えたりしますが、「この得点だよ」と知らせるだけです。

得点表の得点が＋のときは、累計得点が増えますが、－のときは、その分だけ引かれてしまいます。得点表のでき方によっては、－のほうが多くなって、累計得点が減っていくだけになるかもしれません。そうなったら、累計得点が目標に達することは難しくなるでしょう。マス

```
------------------------------
   4   -2    1   -3    1   -2
   7    0   -1    7  *-3    4
  -1    3    3   -5   -3    0
   2   -5    6    4    5    0
   2    3    0    2    2    6
  -3    4   -2    4    0    5
------------------------------
          キミ ノ ﾊﾞﾝ
?0110
 6253   トクテン  -3

          キミ   17   ﾎﾞｸ   26
```

図3－19　点取りゲームの実行例3

```
------------------------------
   4   -2    1   -3    1   -2
   7    0   -1    7   -3    4
  -1  * 3    3   -5   -3    0
   2   -5    6    4    5    0
   2    3    0    2    2    6
  -3    4   -2    4    0    5
------------------------------
          ﾎﾞｸ ノ ﾊﾞﾝ
 0101
 6322   トクテン   3

          キミ  17    ﾎﾞｸ   29
```

図3－20　点取りゲームの実行例4

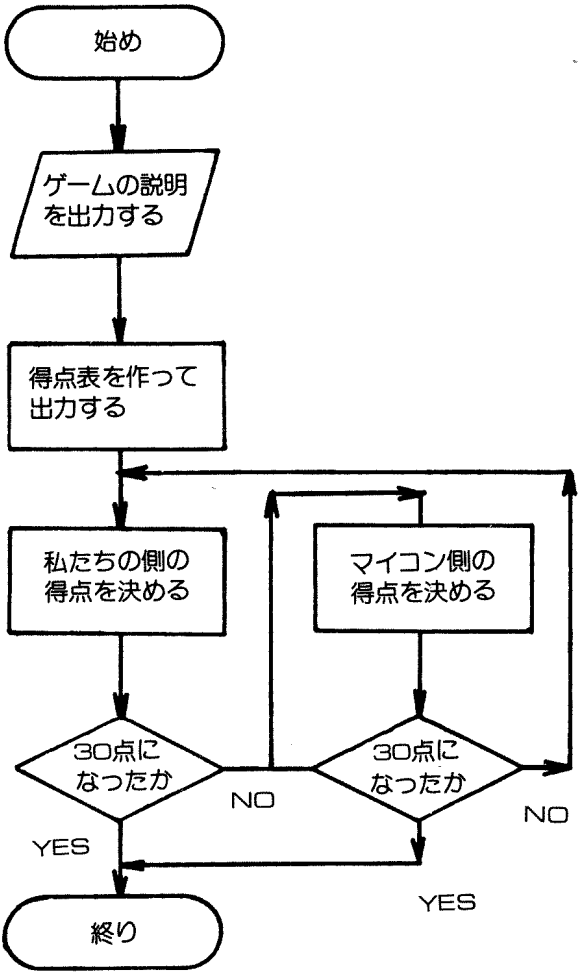


図３－21　点取りゲームの流れ図

クのかけ方が悪くて一になってばかりいるのは、自分の責任です。プログラムが意地悪をすることは絶対にありません。もっとも、RNDで出てくる乱数がひねくれている可能性はあります。

私たちか、マイコンのどちらかの累計得点が30点になると、ゲームはおしまいです。

キミ（ボク）　ノ　カチ!!

というメッセージが出てきます。

点取りゲームのプログラム

プログラムの全体は、図3―22に示しました。メイン・プログラムは、630行までです。640行以下は、サブルーチン（3つ）になっています。

全体のフローチャートは、図3―21です。

図3－22　点取りゲームのプログラム

```
100 REM テントリ ケ゛ーム
110 DIM S (6、6)、F(6)、L(6)、P(2)
120 LET T=30
130 PRINT "テントリ ケ゛ーム"
140 PRINT
150 PRINT "サイコロ ヲ 4コ フリ マス"
160 PRINT "ソノウチ ノ 2コ テ゛ トクテンヒョウ ノ マス"
170 PRINT "ヲ エラヒ゛ マス"
180 PRINT "ト゛ノ 2コ テ゛ エラフ゛ カ ハ マエモッテ"
190 PRINT "キメテ クタ゛サイ"
200 PRINT "2ハ゛ン メ ト 4ハ゛ンメ ノ サイコロ ヲ"
210 PRINT "エラフ゛ トキ ハ ""?"" ノ アト ニ"
220 PRINT """0101"" ト ニュウリョク スレハ゛ ヨイ"
230 PRINT "ノテ゛ス"
240 PRINT "ホ゛ク モ オナシ゛ コト ヲ シマス"
250 PRINT "サキ ニ 30 テン ニ ナッタ ホウ カ゛"
260 PRINT "カチ テ゛ス"
270 FOR I=1 TO 6
```

```
280 FOR J=1 TO 6
290 LET K=INT (RND (1) *13) -5
300 LET S (I、J) =K
310 NEXT J
320 NEXT I
330 CLEAR
340 GOSUB 1050
350 FOR I=1 TO 6
360 FOR J=1 TO 6
370 PRINT TAB (J*4-2);S (I、J);
380 NEXT' J
390 PRINT
400 NEXT I
410 PRINT
420 GOSUB 1050
430 LET C1=0
440 LET C2=0
450 PRINT TAB (7); "キミ ノ ﾊﾞﾝ"
460 INPUT M
470 GOSUB 640
480 LET C1=C1+G
490 CURSOR 16、8
500 GOSUB 910
510 CURSOR 12、1
520 PRINT TAB (7); "ﾎﾞｸ ノ ﾊﾞﾝ"
530 LET M=10** (INT (RND (1) *4))
540 LET M1=10** (INT (RND (1) *4))
550 IF M=M1 THEN 540
560 LET M=M+M1
570 PRINT M
580 GOSUB 640
590 LET C2=C2+G
600 CURSOR 16、8
610 GOSUB 910
620 CURSOR 12、1
630 GOTO 450
640 REM GET
650 FOR I=1 TO 4
660 LET F (I) =0
670 LET M2=10** (4-I)
680 IF M<M2 THEN 710
690 LET F (I) =1
700 LET M =M-M2
```

```
 710 NEXT I
 720 LET H=0
 730 FOR I=1 TO 4
 740 LET L (I) =INT (RND (1) *6) +1
 750 LET H=H*10+L (I)
 760 NEXT I
 770 PRINT H;
 780 LET N=0
 790 FOR I=1 TO 4
 800 IF F (I) =0 THEN 830
 810 LET N=N+1
 820 LET P (N) =L (I)
 830 NEXT I
 840 LET G=S (P (1)、P (2))
 850 PRINT "トクテン";G
 860 LET X1=P (1) +2
 870 LET X2=4*P (2) -2
 880 CURSOR X1、X2
 890 PRINT "*";
 900 RETURN
 910 REM SCORE
 920 PRINT "キミ";C2;
 930 PRINT "ホ゛ク";C2;
 940 CURSOR X1、X2
 950 PRINT " "
 960 IF C1>30 THEN 990
 970 IF C2>30 THEN 1020
 980 RETURN
 990 CURSOR 12、8
1000 PRINT "キミ ノ カチ !!"
1010 STOP
1020 CURSOR 12、8
1030 PRINT "ホ゛ク ノ カチ !!"
1040 STOP
1050 REM LINE
1060 FOR I=1 TO 26
1070 PRINT "−";
1080 NEXT I
1090 PRINT
1100 PRINT
1110 RETURN
1120 END
```

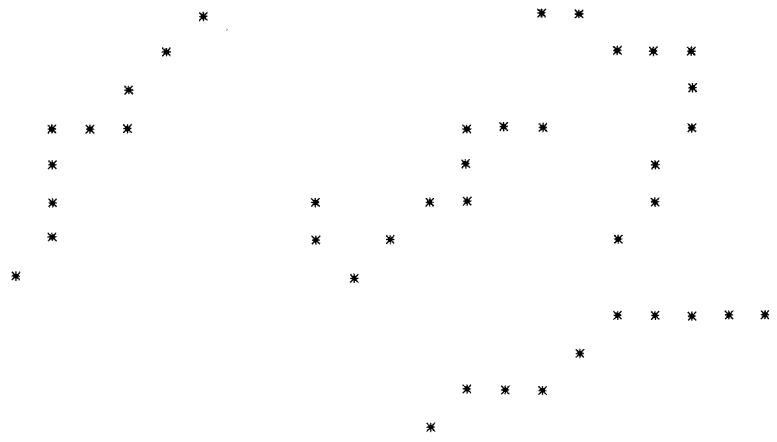


図3－23　画面のムカデ

ムカデ・レース

「ムカデ」は、「百足」と書きます。このムカデに、ゲームの主人公として登場してもらいます。

マイコン・ゲームですから、CRTの画面に、「ムカデらしき図形」を表示します。細かいところまで描くのは、たいへんなので、ここでは、「＊」を10個並べただけのムカデでがまんしましょう。もちろん、10個の「＊」は、つながっていなければなりませんし、また、いつも同じところで、じっとしていては困ります。「＊」が10個つながったまま、画面の上を動き回るのを「ムカデが走り回っている」とみたてるわけです。

最初に、「＊」10個のムカデが画面の上を、あ

ちこち動くだけのプログラムを作ることにします。このプログラムができれば、それを、ムカデ・レースのプログラムに発展させることは簡単です。

プログラムの実行を始めると、画面の中央に突然、ムカデが現われて、いくらかとまどいぎみに動き始めます。画面の端につくと引返しますし、動くのが嫌になっても消えたりはしません。実は、「不可抗力」で動けなくなってしまうことがあるのですが、普通は、かなり長い間、元気に動き回ってくれます（図3—23）。そんなプログラムを作ります。

といっても、初めは見当もつかないかもしれません。

そこで、このゲームに登場するムカデの性質と、その動き方を、もう少し詳しく調べてみましょう。

ムカデ（星）の位置

ムカデつまり、ここでは10個の星（「＊」は、アステリスクといいます。ここでは、「星」と呼んでおきます）が、画面上に適当な形で並ぶためには、星1つ1つの位置をきめてやらなければなりません。画面は、16×32（16×64）の区画に分かれますから、上からX行目、左からY字目、という「XY座標」を使うのが便利です。位置（X、Y）に、星を表示したければ、CURSOR文で、その位置にカーソルを持ってきてから、

PRINT "＊"

を実行します。

星のつながり

ムカデには、頭もあって、10個の星は、お互いにつながっています。それが、ムカデの身体

です。2つの星が「つながっている」というのは、図3―24のように、タテ（X）方向にも、ヨコ（Y）方向にも、1区画しか離れていないことです。位置（X、Y）にある星と、「つながっている」星の位置は、

（X＋1、Y）
（X―1、Y）
（X、Y＋1）
（X、Y―1）
（X＋1、Y＋1）
（X―1、Y＋1）
（X＋1、Y―1）
（X―1、Y―1）

の8つのうちのどれかです。画面上では、

X＋1→下
X―1→上
Y＋1→右
Y―1→左

のような対応があります。たとえば、

（X＋1、Y―1）

は、（X、Y）の「左下」にあたります。

この星のつながりは、ムカデの身体をつくる星を、頭から星1、星2、星3、……、星10と

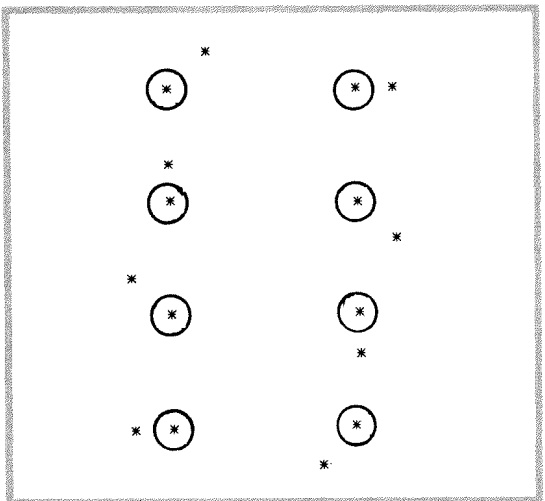


図3－24　ムカデの頭の動き方

したとき、星Nと星N＋1についてだけ考えればよいのです。

つまり、星2と星3は、つながっていなければなりませんが、星4と星6がつながっている必要はありません。星1と星10がつながっていると「輪」になってしまいます(図3―25)。

ムカデの動き

ムカデが1歩すすむためには、足を100本動かさなければならないのでしょうが、そこまで考えると1歩もすすめなくなります。私たちのムカデは、頭だけを動かすことができるのだと考えましょう。頭は、急にどこかへ飛んでいくのではなく、着実に1区画だけすすむのです。

104頁の図3―26にその様子を示しました。ムカデの1歩は、

① 頭が動いた場所に、「＊」を書く

② 尻尾の端の「＊」を消す

という2つのステップから成立っています。①と②を組にして実行しますから、ムカデが何歩歩き回っても、身体の長さは星10個です。

ムカデは、気ままに動いていればよいので、頭の動きは、乱数で決めます。「頭を突込んではいけないところ」もあります。それは、

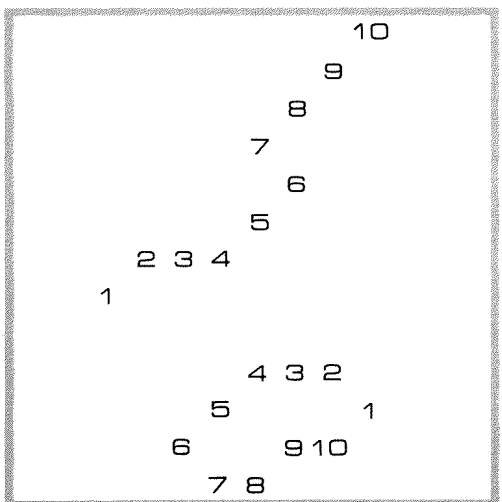


図3－25　頭からの星の順番

① 自分の身体
② 画面の外
の2つです。自分の身体の星に、頭の星を重ねてしまうと奇妙ですし、画面の外にどんどん逃げ出してしまってもいけません。

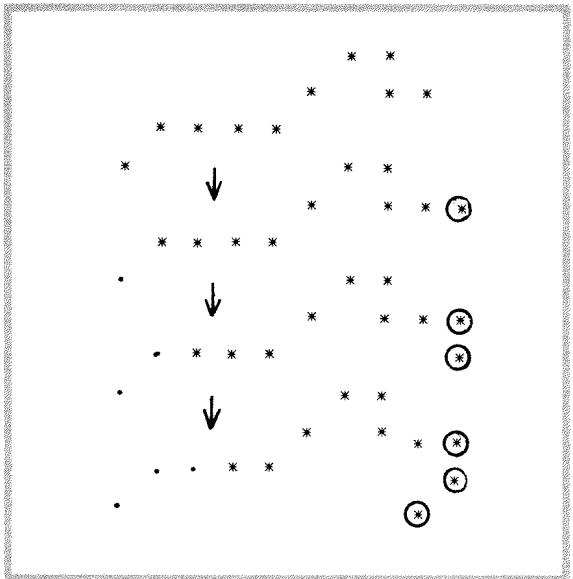


図3－26 ムカデの動き方

①と②の制限のもとで、勝手に動き回ってもらうことにするわけです。

ムカデが動けなくなるとき

実は、動きに関する以上のような制限条件があるために、ムカデが動けなくなってしまうことがあります。

① 画面の角に頭を突込んでしまう
② 自分の身体で、頭を押え込んでしまう

いわば、「とぐろ」をまいたようになるの2つの場合です。図3—27を、見てください。

ムカデの散歩のプログラム

これだけ調べておけば、プログラムを書くこともできるはずです。身体を構成する10個の星のX座標をP(1)、P(2)、……P(10)、Y座標を、

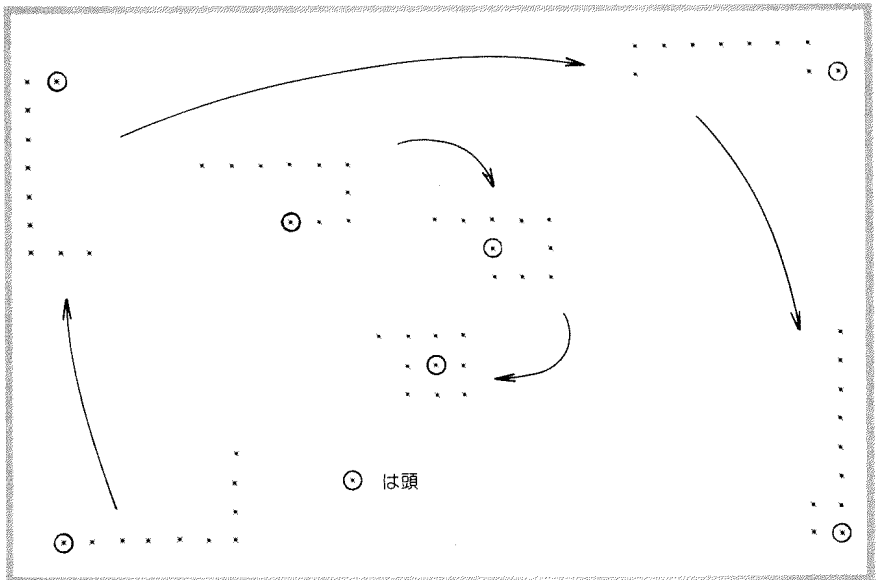


図3－27　ムカデが動けなくなるとき

Q(1)、Q(2)、……、Q(10)という配列変数に入れておきます。「頭」の位置は、変数Nを使って、(P(N)、Q(N))とします。ムカデが1歩すすむたびに、Nを1増加させ、新しい頭の位置を、P(N)とQ(N)に代入するのです。Nは、11になるところで、1に戻されます。つまり、P、Qを「輪」のように使うわけです(106頁図3—28)。

「ムカデの散歩」のプログラムを107頁図3—29、フローチャートを108頁の図3—30に示しました。

図3—29の300～390が、1歩すすむ、という部分です。尻尾を消して、頭を書いていることに注意してください。

図3—29のプログラムでは、XとYがともに変化する動き方、つまり、頭が斜めの方向に動く動き方を許しています。ですから、ときには、奇妙なことも起こります。図3—31では、自分の身体で結び目をつくってしまうのですが、平

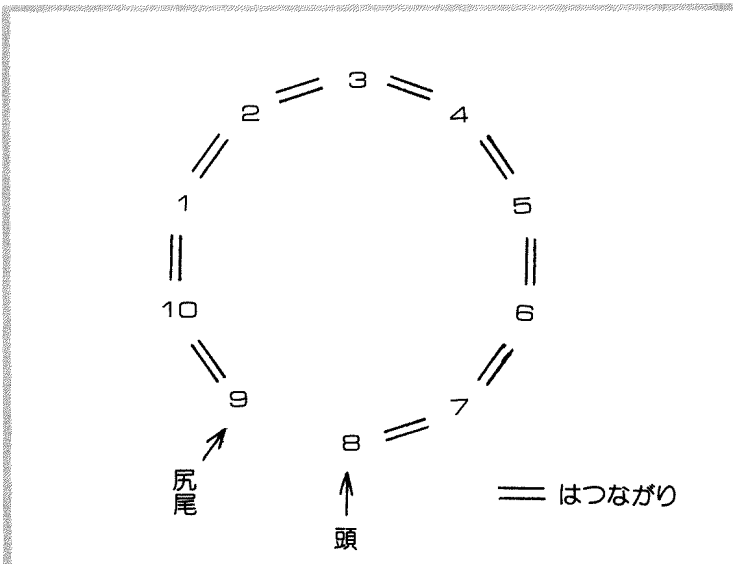


図3－28　配列P・Qの使い方

気な顔で、すすんでいきます。

ムカデ・レースのプログラム

これでやっと、ムカデ・レースのプログラムを書く準備ができました。ムカデ・レースといっても、実は、画面を3つに区切って、その中で、それぞれムカデの散歩をさせようというだけです。

ただ、スタートとゴールは、一応ありますから、ムカデの動きもランダムにするのではなく、ゴールに向うような動きにします。

もちろん、ムカデらしさを残すために、あちこち動き回る余裕はとっておきましょう。

プログラムは110頁の図3―32です。このプログラムによるムカデの動きが図3―29のプログラムと違う点は、頭が斜めに動くことを許さないことです。「前に進む」（画面では右に進む）

```
100 REM CENTIPEDE
110 DIM P(10)、Q(10)
120 CLEAR
130 FOR I=1 TO 10
140 LET P(I) =8
150 LET Q(I) =8+I
160 NEXT I
170 CURSOR 8、9
180 PRINT "**********"
190 LET N=10
200 LET X=P(N) +INT(RND(1) *3) -1
210 IF X<1 THEN 200
220 IF X>16 THEN 200
230 LET Y=Q(N) +INT(RND(1) *3) -1
240 IF Y<1 THEN 230
250 IF Y>32 THEN 230
260 FOR I=1 TO 10
270 IF X<>P(I) THEN 290
280 IF Y=Q(I) THEN 200
290 NEXT I
300 CURSOR X、Y
310 PRINT "*"
320 LET N=N+1
330 IF N=11 THEN LET N=1
340 LET A=P(N)
350 LET B=Q(N)
360 CURSOR A、B
370 PRINT " "
380 LET P(N) =X
390 LET Q(N) =Y
400 GOTO 200
410 END
```

図3－29 ムカデのプログラム

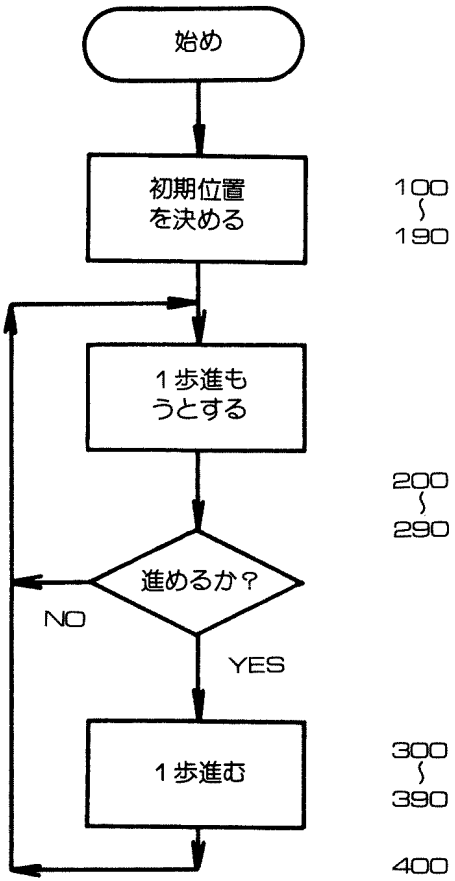


図3－30　ムカデのフローチャート

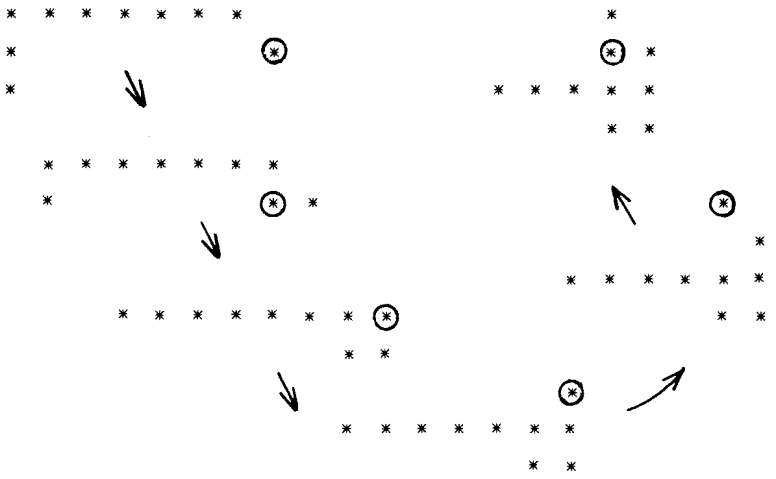


図3－31　ムカデの変則的な動き方

確率を大きくしてあるので、「とぐろ」をまいて動かなくなる確率も小さいでしょう。

プログラムの応用

実は、図3―32のプログラムは、「見ているだけ」のプログラムです。プログラムの実行を始めると、マイコンの方で勝手に、ムカデを走らせ始め、1匹がゴールインすると、そこで実行が終ります。

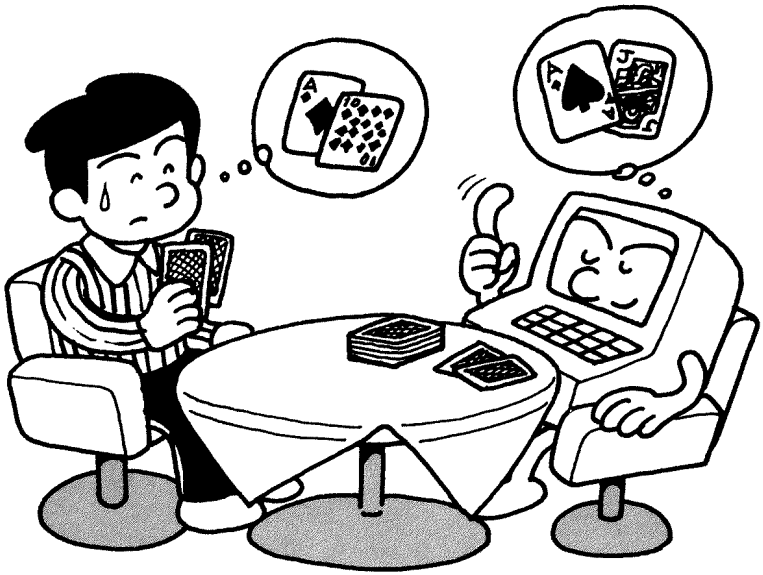
また、星10個でムカデといいましたが、長いものなら何にみたててもよいわけです。星3個くらいに短くすれば、ほとんど、すべての「生き物」にみたてることができます。

図3－32　ムカデ・レースのプログラム

```
100 REM RACE
110 CLEAR
120 PRINT "ﾑｶﾃﾞ　ﾚｰｽ
130 PRINT
140 PRINT "ﾑｶﾃﾞ　ﾊ　3
    ﾋﾞ　ｷ　ｲﾏｽ"
150 PRINT
160 PRINT "ﾋﾀﾞﾘ　ｶﾞﾜ
    ｶﾗ　ｽﾀｰﾄ　ｼﾃ　ﾐｷﾞ
    ﾊｼ"
170 PRINT "ﾆ　ﾂｸﾄ
    ｺﾞｰﾙｲﾝ　ﾃﾞｽ"
180 PRINT "ﾄﾁｭｳ　ﾃﾞ
    ｳｺﾞｶﾅｸ　ﾅﾙ　ﾄ
    ｼｯｶｸ"
190 PRINT "ﾆ　ﾅﾘﾏｽ"
200 PRINT
210 PRINT "ｿﾚﾃﾞﾊ
    ｽﾀｰﾄ!!"
220 CLEAR
230 FOR I=1 TO 3
240 GOSUB 1110
250 FOR J=1 TO 4
260 PRINT I;TAB
    (30);I
270 NEXT J
280 NEXT I
290 GOSUB 1110
300 DIM P(3,10), Q
    (3,10), S(3)
310 FOR I=1 TO 3
320 LET K=5*(I-1)
    +1
330 LET P (I, 1) =
    K+2
340 LET P (I, 2) =
    K+1
350 LET P (I, 3) =
    K+1
360 LET P (I, 4) =
    K+2
370 LET P (I, 5) =
    K+3
380 LET P (I, 6) =
    K+4
390 LET P (I, 7) =
    K+4
400 LET P (I, 8) =
    K+3
410 LET P (I, 9) =
    K+3
420 LET P (I, 10) =
    K+3
430 LET Q (I, 1) =4
440 LET Q (I, 2) =4
450 LET Q (I, 3) =3
460 LET Q (I, 4) =3
470 LET Q (I, 5) =3
480 LET Q (I, 6) =3
490 LET Q (I, 7) =4
500 LET Q (I, 8) =4
510 LET Q (I, 9) =5
520 LET Q (I, 10) =6
530 NEXT I
540 FOR I=1 TO 3
550 FOR J=1 TO 10
560 CURSOR P (I), Q (J)
570 PRINT "*";
580 NEXT J
590 LET S (I) =0
600 NEXT I
610 FOR C=1 TO 3
620 GOSUB 670
630 IF W=1 THEN 660
640 NEXT C
650 GOTO 610
660 STOP
670 REM STEP
680 LET E=5*C-3
690 LET F=5*C
700 LET N=0
710 LET N=N+1
```

```
 960 CURSOR X, Y
 970 PRINT "*";
 980 LET S (C) =
     S (C) +1
 990 IF S (C) <11
     THEN 1010
1000 LET S (C) =1
1010 LET A=
     P (C, S (C))
1020 LET B=
     Q (C, S (C))
1030 CURSOR A, B
1040 PRINT " ";
1050 LET P (C, S (C)) =
     X
1060 LET Q (C, S (C)) =
     Y
1070 LET W=0
1080 IF Y<30 THEN
     1100
1090 LET W=1
1100 RETURN
1110 REM LINE
1120 PRINT " ";
1130 FOR K=1 TO 30
1140 PRINT "-";
1150 NEXT K
1160 RETURN
1170 END
```

```
 720 IF N<10 THEN
     740
 730 RETURN
 740 LET T=RND (1)
 750 IF T>0.2 THEN
     790
 760 LET X=
     P (C, S (C)) -1
 770 LET Y=
     Q (C, S (C))
 780 GOTO 890
 790 IF T>0.4
     THEN 830
 800 LET X=
     P (C, S (C)) +1
 810 LET Y=
     Q (C, S (C))
 820 GOTO 890
 830 IF T>0.6
     THEN 870
 840 LET X=
     P (C, S (C))
 850 LET Y=
     Q (C, S (C)) -1
 860 GOTO 890
 870 LET X=
     P (C, S (C))
 880 LET Y=
     Q (C, S (C)) +1
 890 IF X<E THEN
     710
 900 IF X>F THEN
     710
 910 IF Y<3 THEN
     710
 920 FOR I=1 TO 10
 930 IF X<>P (I, S (I))
     THEN 950
 940 IF Y=Q (I, S (I))
     THEN 710
 950 NEXT I
```



ブラック・ジャック

52枚のカードを使ったトランプ・ゲームには、数えきれないくらい多くの種類があります。ブリッジ、ポーカー、ラミーなどは、代表的なゲームです。遊び方には、少し頭を使うものか、まったく運に頼るものか、という違いがあります。ここに取上げたブラック・ジャックは「運に頼るゲーム」です。ラスベガスやモンテカルロのカジノでは、ポーカーと並んで人気のあるゲームになっているそうです。

カード・ゲームのルール

ルールは、親（ディーラー）が、52枚のカードをよく切ってから子（プレイヤー）に1枚ずつ配ります。プレイヤーは、まだカードをみて

はいけません。ディーラーは、プレイヤーにカードを配ったあと、自分の手もとにもやはり伏せたまま1枚配ります。

次にディーラーは、プレイヤーの1人に、カードを1枚、今度は表向きで配ります。カードを配られたプレイヤーは、手もとに伏せてあったカードを、このときになって初めてみることができます。

プレイヤーは、2枚のカードをみて、もう1枚カードをもらうかどうかを、決めなければなりません。このときの目標は、カードを合わせて21点にすることです。2から10までのカードは、その数の通りの点、ジャック、クィーン、キング（フェイス・カード）は10点、エースは1点か11点です。たとえば、2枚のカードがハートのクィーンと、クラブの4の場合は、クィーンが10点ですから、14点になります。21点には、7点足りないわけです。

もう1枚カードをもらって、もらったカードが、ダイヤのエースだったとしましょう。エースを1点と考えるか、11点と考えるかは、プレイヤーが、その場に応じて決めてかまいません。11点と考えると、手もとにあった14点と合わせて25点になります。21点を越えてしまうと、「ダウン」とか「ドボン」とかいって失格です。ここでは1点と考えて、手もとにあった2枚のカードと合わせて15点としましょう。あと6点ですが、もう1枚ひくかどうかは難しいところです。21点になるまでは、何枚でも、もらうことができますから、冒険ですが、もう1枚もらってみましょう。

もらったカードが、たまたまハートの6だったとします。そうすると、手もとの3枚が15点ですから、合わせて21点になります。21点にな

ったことを「ブラック・ジャック」といいます。「ブラック・ジャック」は一番強いので、これ以上カードをひく必要はありません。

21点になるまでカードをひきつづけるという手順を各プレイヤーが済ませたあと、ディーラーも、同じように、21点に近くなるようにカードをひいていきます。勝負は、プレイヤー一人ずつと、ディーラーの勝負です。どちらかが、21点のブラック・ジャックなら、ブラック・ジャックを作った方が勝ちですし、どちらも21点より少なければ、1点でも21点に近い方が勝ちです。21点を越えた場合は、負けです。

マイコン・ゲームの遊び方

このブラック・ジャックのゲームをマイコンで楽しんでみましょう。マイコンがディーラーになって、私たちにカードを配ります。カードを配るといってもマイコンが手を出すわけにはいきませんから、CRT上に「スペード・6」のように表示することにします。私たちにハートのクィーンとクラブの4が配られたときは、画面上に

ンート　Q
ラブ　4

のような表示が出ます。マイコンがディーラーですから、自分にもカードを配ります。スペードのA

アナタ (ﾌﾟﾚｲﾔｰ)		マイコン (ﾃﾞｨｰﾗｰ)	
ハート	Q	スﾍﾟｰﾄﾞ	A
クラﾌﾞ	4	クラﾌﾞ	6

図3－33　カードの表示

とクラブの6なら

スペード A
クラブ 6

という表示が画面に出ます。たとえば、図3—33のような画面が、「カードを配る」ときの画面です。

```
ﾌﾞﾗｯｸ ｼﾞｬｯｸ

ｱﾅﾀ ﾉ ﾓﾁﾃﾝ ﾊ 1000 ﾃﾝ ﾃﾞｽ
ﾜﾀｼ ｶﾞ ﾃﾞｨｰﾗｰ ﾖ ﾂﾄﾒﾏｽ

ﾌﾞﾗｯｸ ｼﾞｬｯｸ ｶﾞ ﾃﾞﾙ ﾄ ｶｹﾃﾝ
ﾊ 2 ﾊﾞｲ ﾆ ｱﾂｶﾜﾚﾏｽ
```

図3—34 ブラック・ジャックの実行例1

ゲームの手順

プログラムの実行を始めると、図3—34のような表示が出てきます。

```
        ｱﾅﾀ                ﾜﾀｼ

ｽﾍﾟｰﾄﾞ  Q          ﾊｰﾄ        J

ﾅﾝﾃﾝ ｶｹﾏｽｶ

?200                    ｶｹﾃﾝ   200
                        ﾓﾁﾃﾝ   1000
```

図3—35 ブラック・ジャックの実行例2

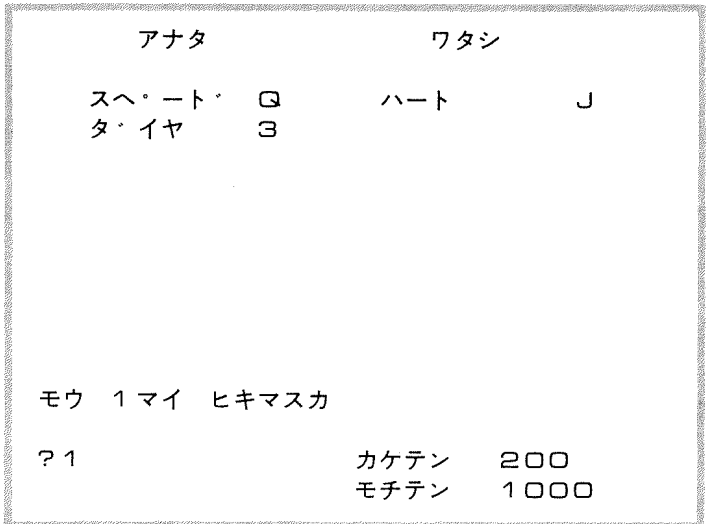


図3－36 ブラック・ジャックの実行例3

次に、図3—35のような画面に変ります。マイコンと私たちにカードが1枚ずつ配られた状態です。私たちの最初の持点は1000点です。

ナンテン カケマスカ

と出てきたら、「?」のあとに賭ける点を入力します。200点賭けるなら、「200」と入力すればよいのです。そうすると、

カケテン 200

のように画面の右下に表示されます。これで、ゲームを始める準備ができました。カード・ゲームでは、1枚目に配られたカードは裏向きでしたが、マイコン・ゲームでは、初めから表向きです。「マイコンがインチキをするのではないか?」と疑ってはいけません。

最初は、私たちがカードをひく番です。まず2枚目のカードが、1枚目に配られたカードの下に出てきます。配られた2枚のカードをみて、

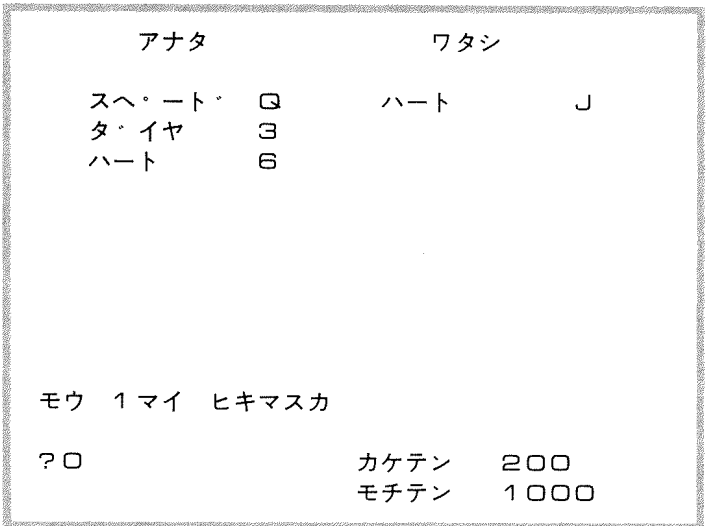


図3－37 ブラック・ジャックの実行例4

もう1枚ひくかどうかを決めてください。

モウ 1マイ ヒキマスカ

と聞いてきますから、もう1枚ほしいとき（HIT）は「1」、もういらないとき（STAND）は、「0」を入力します。なおカードの表示で、エース、ジャック、クィーン、キングはそれぞれA、J、Q、Kのように略記されます。

図3—36と図3—37に、私たちがカードをひくときの画面の変化を示しました。私たちに配られたカードは、スペードのQ、ダイヤの3、ハートの6ですから、19点になります。もう1枚ひくと21点を越えてしまいそうですから、ここでやめておきましょう。

マイコンが「モウ 1マイ ヒキマスカ」と聞いてきたとき、私たちが「0」を入力すると、マイコンは、自分のカードをひき始めます。このとき、マイコンは、

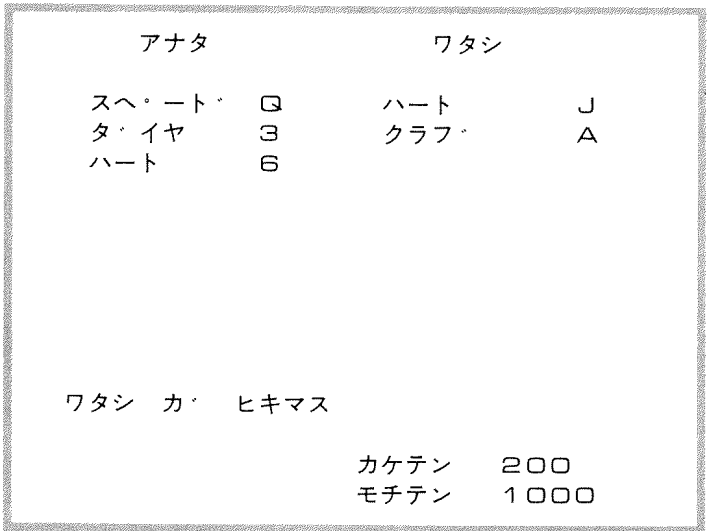


図3－38 ブラック・ジャックの実行例5

ワタシ カ ヒキマス

と、私たちに知らせてくるはずです。マイコンは、自分の判断で、21点に近くなるように、そして21点を越えないように、カードをひいていきます。マイコンは、自分のカードをひくときには、私たちの「成績」を知りません。私たちが「ダウン」であっても、マイコンは、手を抜いたりせず、自分の手が少しでも21点に近くなるように努力します。

マイコンが、カードをひき終ると、私たちとマイコンの成績が表示されます。19点なら、

トータル 19

のように出てくるのです。21点なら、

ブラック ジャック !!

と出ますし、21点を越えてしまうと、

ダウン !!

と表示されます。そこで私たちとマイコンとの

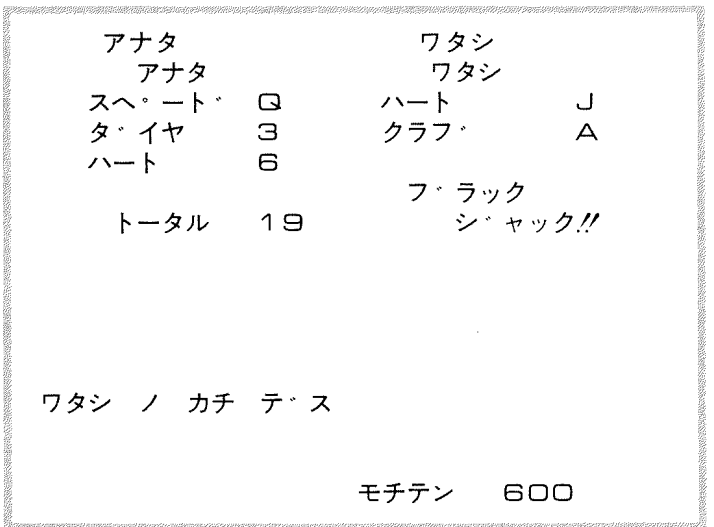


図3－39 ブラック・ジャックの実行例6

勝負が決まりますから、

ワタシ ノ カチ デス
アナタ ノ カチ デス

のようなメッセージが出てきます。

図3―38が、マイコンのひいているときの画面です。図3―39で、勝負が決まります。

勝負が決まると同時に、賭点の精算が行われ私たちの持点が増えたり減ったりします。どちらかがブラック・ジャックの場合は、賭点が2倍になって増えたり減ったりします。

プログラムの説明

プログラム全体の説明に入る前に、マイコンでトランプ・ゲームを扱うための基本的手法について説明しておきましょう。これから紹介するいくつかの手法は、ブラック・ジャックに限らず、ほかのトランプ・ゲームのプログラムに

も応用することができます。

カードの表示

CRT上に特定のカードを表示するために、「スペード」とか、「Q」のような文字を使います。これらの文字は、

PRINT "スペード Q"

のような文で画面に出力することもできますがカードごとにPRINT文を変えるのは、たいへんです（52枚のカードに対して、52のPRINT文が必要になります！）。そこで、カードの種類（SUIT）4つと、カードの位（KIND）13を組合わせて、特定のカードを画面に出力するようにします。出力する文字を、文字型の配列変数に入れておけば、使いやすいでしょう。図3－40が、文字型の配列変数Q$に、カードの種類と位を表わす文字を代入するプログラムです。このプログラムを実行すると、

```
100 REM CARDS
110 DIM D (52), Q$ (17)
120 FOR I=1 TO 17
130 READ Q$ (I)
140 NEXT I
150 DATA "スペード", "ハート"
160 DATA "ダイヤ", "クラブ"
170 DATA "A", "2", "3", "4", "5"
180 DATA "6", "7", "8", "9", "10"
190 DATA "J", "Q", "K"
```

図3－40　カード表示のためのプログラム

Q$(1) は "スペード" という文字
Q$(2) は "ハート" という文字
……
Q$(16) は "Q" という文字
Q$(17) は "K" という文字

というように、Q$の値が決まります。

スペード Q

と出力したければ、

PRINT Q$(1);Q$(16)

という文を実行させればよいのです。

ハート 6

と出力するには、

PRINT Q$(2);Q$(10)

を実行する必要があることに注意してください。文字の「6」は、Q$(10)という変数に入っています。カードの位の数（A、2、3、4……）と、Q$の添数（Q$(N)と書いたときのN）とは、直接には対応しません。

ここでは、カードの種類を表わすために、「スペード」とか「ハート」というカナ文字を使いましたが、マイコンによっては、

のような記号が使えます。そのような記号が使えるときは、使うようにしてください。ゲームがずっと楽しくなるはずです。そのためには、図3—40のプログラムの150行と160行のDATA文を、

150 DATA "♠", "♥"
160 DATA "♦", "♣"

のように変えなければなりません。

カードの番号づけ

プログラムで、それぞれのカードを区別する

```
2000 REM SHUFFLE
2010 FOR I=1 TO 52
2020 LET D(I)=0
2030 NEXT I
2040 FOR I=1 TO 52
2050 LET J=INT(RND(1)*52)+1
2060 IF D(J)<>0 THEN 2050
2070 LET D(J)=I
2080 NEXT I
2090 LET N=0
2100 RETURN
```

図3－41　カードをシャッフルするサブルーチン

ために、52枚のカードに1から52までの番号をつけます。スペードAが1、スペード2が2、クラブQが51、クラブKが52です。つまり、52枚のカードを、スペード、ハート、ダイヤ、クラブの順にまとめ、それぞれの種類の中でAからKの順に並べたと考えて、最初から番号をつけていくわけです。

カードのシャッフル

52枚のカードをよく切りまぜることを、シャッフルといいます。プログラムの中では、D(52)という配列変数の中に、1から52までの数字をデタラメな順序で代入していくことで、シャッフルを行います。D(1)が1枚目のカード、D(2)が2枚目のカードを表わしますから、

D(1)＝11、D(2)＝34

なら、1枚目のカードがスペードJ、2枚目のカードがダイヤ8ということになります。もち

ろん、D(1)、D(2)……、D(52)の中に同じ数があっては困ります（D(14)＝39、D(23)＝39なら、14枚目も23枚目もダイヤKです！）。

シャッフルのプログラムを図3—41に示しました。2000番の文番号から始まるサブルーチンにしてあります。

カード番号と種類・位との対応

52枚のカードを1から52までの数で表わすことにしましたが、画面上にその数をそのまま出力しても何のことだかわかりません。そのために、

39→ダイヤK
11→スペードJ

のように変換するプログラムが必要になります。このプログラムもサブルーチンにしました。図3—42にあげておきます。

図3—40～42のプログラムは、カード・ゲー

```
2110 REM DEAL
2120 LET N=N+1
2130 LET C=D(N)
2140 LET S=INT((C-1)/13)+1
2150 LET T$=Q$(S)
2160 LET K=C-13*S+13
2170 LET L$=Q$(K+4)
2180 RETURN
2190 END
```

図3－42　カードを1枚ひくサブルーチン

```
200 REM EXAMPLE
210 GOSUB 2000
220 FOR I=1 TO 52
230 GOSUB 2110
240 PRINT T$;TAB (7) ;L$
250 NEXT I
260 STOP
```

図3－43　カード・シャッフルのプログラム

ムのプログラムの基本的な手法として、ほかのプログラムにも応用できるものです。

「カードをよくきって1枚目から表示していく」というプログラムを試してみましょう。図3―40～42のプログラムと、図3―43のプログラムを合わせてマイコンに入力し、「RUN」と入力すれば、図3―44のように、52枚のカードが次々に表示されていきます。

ブラック・ジャックのプログラム

プログラムの残りは、126頁の図3―45に示しました。図3―40～42のプログラムと、図3―45のプログラムを合わせれば、完全なブラック・ジャックのプログラムです。

図3―45のプログラムについて、簡単に説明してみましょう。

200～300行　最初に、ゲームの説明を画面に出

力します。300行では、私たちの持点を1000点にしています。

310～410行　画面を新しくして、ゲームを始める準備をします。いくつかの変数に最初の値を代入し、カードもシャッフルします（410行で、カードをシャッフルするサブルーチンを呼んでいます）。

420～530行　私たち（画面上では「アナタ」）と、マイコン（画面上では「ワタシ」）に、カードを1枚ずつ配って、その結果を画面に表示します。

540～650行　私たちに、カードをもう1枚配り、もっと必要かどうかを聞いてきます。必要なら次のカードを配ってくれます。

660～750行　今度はマイコンのひく番です。手もとのカードが合わせて21点に近くなるように、そして21点を越えないように考えてカードをひいていきます。

760～950行　私たちとマイコンの点を表示します。21点なら「ブラック・ジャック!!」と表示しますし、21点を越えてしまうと「ダウン!!」と表示します。

960～1100行　勝負を決めます。賭点にしたがって、私たちの持点を増やしたり減らしたりするのもこの部分です。

1110～1250行　カードの点を決めるために使うサ

```
ハート　　A
スペード　　6
ハート　　4
クラブ　　J
クラブ　　8
ハート　　5
ダイヤ　　K
ハート　　J
……
```

図3－44　シャッフルされたカード表示

ブルーチンです。

マイコンが自分の戦法を決めるのは、次のような規則によっています。

(1) 手もちのカードの点が11点以下なら、必ずもう1枚ひく。

(2) 手もちのカードが12点以上、20点以下のときは、もう1枚ひくときもひかないときもある。もちろん、21点に近いほど、もう1枚ひく可能性は小さくする。

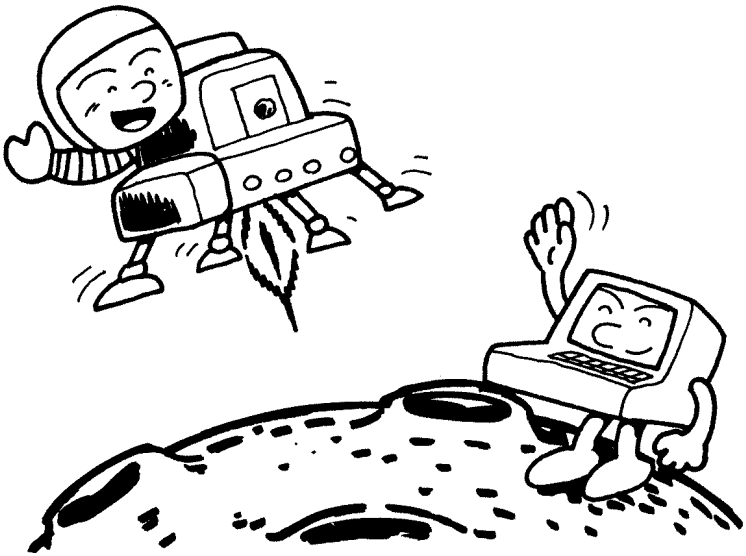
(3) 手もちのカードが21点以上なら、そこで、やめる。

結局、マイコンも、人間と同じことを考えているわけです。ただ、人間の場合と違って、マイコンの「思考」は、プログラムの形になっているということです。

図3－45 ブラック・ジャックのプログラム

```
200 REM BLACK JACK
210 DIM A(2),
        H(2),N(2)
220 CLEAR
230 PRINT "ﾌﾞﾗｯｸ
        ｼﾞｬｯｸ"
240 PRINT
250 PRINT "アナタ ノ
        モチテン ハ 1000
        テン ﾃﾞｽ"
260 PRINT "ワタシ ｶﾞ
        ﾃﾞｨｰﾗｰ ヲ ツトメ
        マス"
270 PRINT
280 PRINT "ﾌﾞﾗｯｸ
        ｼﾞｬｯｸ ｶﾞ ﾃﾞﾙ
        ﾄ カケテン"
290 PRINT "ﾊ 2 ﾊﾞｲ
        ニ アツカワレ マス"
300 LET E=1000
310 CLEAR
320 PRINT TAB(4)
        ;"アナタ";
330 PRINT TAB
        (16);"ワタシ";
340 CURSOR 16,14
350 PRINT "モチテン";E
360 FOR I=1 TO 2
370 LET A(I)=1
380 LET H(I)=0
390 LET N(I)=2
400 NEXT I
410 GOSUB 2000
420 GOSUB 2110
430 LET M=1
440 GOSUB 1110
450 GOSUB 2110
460 LET M=2
470 GOSUB 1110
480 CURSOR 13,1
490 PRINT "ナンテン
        カケマス カ"
```

```
 880 PRINT "フ゛ラック"
 890 LET X=X+1
 900 CURSOR X, Y
 910 PRINT
          "  シ゛ャック  !!"
 920 GOTO 950
 930 PRINT "タ゛ウン  !!"
 940 LET H (M) =0
 950 NEXT M
 960 CURSOR 13, 1
 970 IF H (1) =H (2)
                THEN 1090
 980 IF H (1) >H (2)
                THEN 1040
 990 LET E=E-B
1000 IF H (2) <21
                THEN 1020
1010 LET E=E-B
1020 PRINT "ワタシ
          ノ カチ テ゛ス"
1030 GOTO 310
1040 LET E=E+B
1050 IF H (1) <21
                THEN 1070
1060 LET E=E+B
1070 PRINT "アナタ
          ノ カチ テ゛ス"
1080 GOTO 310
1090 PRINT "ヒキワケ
                テ゛ス"
1100 GOTO 310
1110 REM CHECK
1120 LET N (M) =N (M) +1
1130 LET X=N (M)
1140 LET Y=12*M-9
1150 CURSOR X, Y
1160 PRINT T$
1170 LET Y=Y+7
1180 CURSOR X, Y
1190 PRINT L$
1200 IF K<10
                THEN 1220
1210 LET K=10
1220 LET H (M) =H (M)+K
1230 IF K>1
                THEN 1250
1240 LET A (M) =0
1250 RETURN
 500 PRINT
 510 INPUT B
 520 CURSOR 15, 14
 530 PRINT "カケテン";B
 540 GOSUB 2110
 550 LET M=1
 560 GOSUB 1110
 570 CURSOR 13, 1
 580 PRINT "モウ 1
          マイ ヒキマス カ"
 590 PRINT
 600 INPUT R
 610 IF R=0 THEN 660
 620 GOSUB 2110
 630 LET M=1
 640 GOSUB 1110
 650 GOTO 570
 660 CURSOR 13, 1
 670 PRINT "ワタシ カ゛
                ヒキマス"
 680 GOSUB 2110
 690 LET M=2
 700 GOSUB 1110
 710 LET P=H (2) +
          10*RND (1) -21
 720 IF A (2) >0
                THEN 750
 730 IF H (2) >11
                THEN 750
 740 LET P=P+10
 750 IF P<0 THEN 680
 760 FOR M=1 TO 2
 770 LET X=N (M) +2
 780 LET Y=12*M-8
 790 CURSOR X, Y
 800 IF H (M) >21
                THEN 930
 810 IF H (M) =21
                THEN 880
 820 IF H (M) >11
                THEN 860
 830 IF A (M) >0
                THEN 860
 840 LET H (M) =
                H (M) +10
 850 GOTO 810
 860 PRINT "トータル"
                ;H (M)
 870 GOTO 950
```



月着陸ゲーム

この「月着陸ゲーム」は、ある高さの宇宙空間にいる宇宙船をあやつって、無事に月面に着陸させるゲームです。

月には引力がありますから、何もしないでいれば、宇宙船はどんどんスピードを増していって、最後には、月面に激突してしまいます。そうすれば、宇宙船がこわれてしまうだけでなく、中にいる乗組員も死んでしまいますから、月面に到達した時には、速度が0になるようにしなければなりません。そのとき、あなたが操作できるのは、燃料の噴射量だけです。

燃料を多く噴射すれば、スピードは減ります。いつ、この燃料を噴射すればよいのか、あまり早く燃料を使ってしまうと、燃料には限りがあ

りますから、月に着陸しないうちに燃料ギレになってしまうかもしれません。燃料ギレになってしまったら、あとはどうすることもできません。引力にまかせて、宇宙船が月に激突するのをみるだけです。

プログラムの仕組み

このプログラムを書くためには、まず、いくつかのことを定めておかなければなりません。宇宙船が、ある瞬間(T)に、どのような状態であるかを知るためには最初のときの高度(H)、スピード(V)、燃料の量(F)がわからなくてはいけません。また、燃料をどれくらい使うと、スピードがどれくらい減り、また、高度がどれくらい変化するか、その間の関係も知る必要があります。

以上のことは、ある程度、適当でよいのですが、せっかく、このようにプログラムを作るのですから、正しく物理学の法則にかなったものにしたいと思います。

時間の決め方

時間(T)は、1秒ごとに区切って考えることにします。

速度の決め方

速度(V)は、燃料を噴射しないで、月の引力に引かれるままにしたときに、毎秒2キロメートルづつ、加速されていくことにします。

燃料と速度

この速度を減少させるために、燃料を使うのですが、ここでは、燃料を1キロリットル噴射すると、落下速度は、0・5キロメートル毎秒減少するものとします。なお、符号としては、Fは、現在残っている燃料の量、Wは、ある瞬間に噴出する燃料の量を表わしています。この

T：時間→1秒ごと

V：速度→毎秒2Kmづつ加速される

H：高度→落下距離は、ある瞬間の速度V_1と

1秒後の速さV_2の平均速度×経過時間

$$落下距離=\frac{V_1+V_2}{2}\times 1$$

$$したがって\quad H\leftarrow H-\frac{V_1+V_2}{2}$$

F：残っている燃料の量→1回に噴射する燃料の量をWとすること

$F\leftarrow F-W$　　ただし、$0\leqq W\leqq 30$の範囲に限る

初期値（ゲーム開始時の値）

T＝0, V＝40Km/S（月面に向かって落下中）

H＝200Km, F＝150Kl

図3－46　ゲームのきまり

Wは、それぞれの瞬間ごとに、皆さんが入力していくものです。もちろんWは、現在残っている燃料の量Fを超えてはいけませんし、また、今までに入力したWのすべての合計もF_0（最初のときに宇宙船が持っていた燃料の量）を超えてもいけません。もし、Wの合計がF_0を超えると、「燃料ぎれ」となり、以下は、宇宙空間をただようことになります。

ある瞬間（T_1）から1秒後（T_2）の間の高さの変化は、物理の公式通り、T_1とT_2の間の平均速度×1秒で求められます。ここで平均速度はT_1とT_2の速度を合計して2で割って求めることにします。

以上の約束を表にしたものが、図3－46に示してあります。なお、ゲーム開始の時点で、宇宙船は、200キロメートルの高度から、毎秒40キロメートルの速度で、月面に向かって落

下している途中とします。燃料は、全部で150キロリットル積んでいます。

プログラムの流れ

以上のようなきまりを定めてしまえば、このプログラムもほぼ完成したようなものです。プログラムの概要は、図3－47のあらいフローチャートで示したように、

① 初期値を設定し、
② 噴射する燃料の量Wを入力し、
③ 速度、燃料量、高度の計算をする。
④ そして、高度が0（月面に到達）、あるいは、0以下（月面に激突）ならば、ゲーム終了。

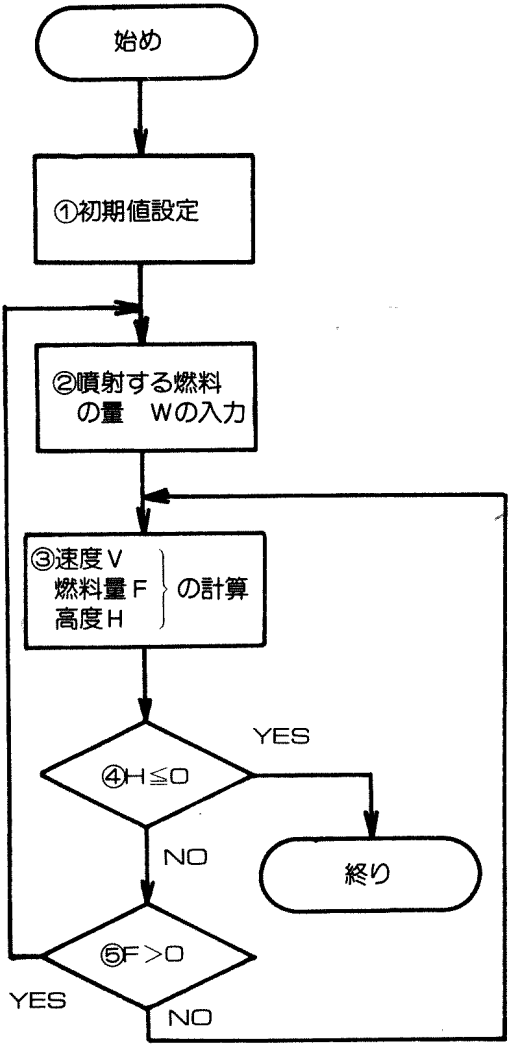


図3－47　あらいフローチャート

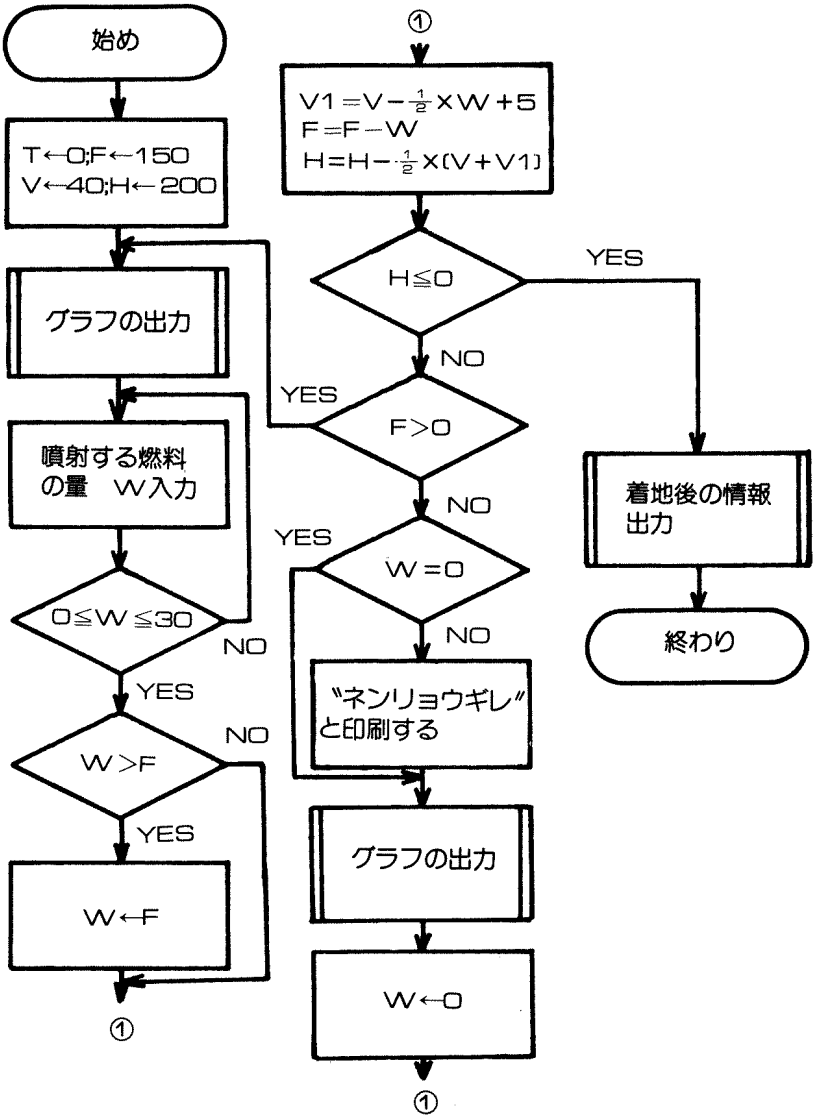


図3－48　くわしい流れ図

⑤ 高度が0より大きいときには、燃料が残っていれば②にもどり、次の瞬間の噴射量を入力し、以上をくりかえす。しかし、燃料が残っていなければ②の入力を省略し、③から以上の手続きをくり返す。

この過程で、適当なところで、宇宙船の位置などの出力が必要です。

プログラムの概要がつかめたら、次は、それをプログラミングするために、もう少しくわしい流れ図を書く必要があります。

それが、図3—48の流れ図です。

プログラムの各部分

この流れ図をそのままプログラムに書くと、134頁の図3—49のプログラムとなります。

このプログラムでは、グラフを書く部分を、800行以下のサブルーチンにしています。

ゲームの遊び方

プログラムの実行を始めると、まず、135頁の図3—50のような画面が出てきます。最初は、

T＝40　(時刻 0)
V＝40　(速度毎秒 40キロメートル)
H＝200　(高度 200キロメートル)
F＝150　(燃料 150キロリットル)

になっています。

I　　+

というグラフは、月面(I)と、宇宙船(+)の距離を表わしています。1字分のスペースが、20キロメートルです(135頁の図3—51)。

「?」に対して、噴射量を入力します。噴射量は、0から30までで、それ以上の数を入力すると、

フンシャリョウ ン 0 カラ 30 マデ

```
320 IF W=10 THEN 350
330 LET T1=
    (-V+SQR(V*V+H*
      (4-W)))/(2-0.5*W)
340 GOTO 360
350 LET T1=H/V
360 LET V1=V+
          (2-0.5*W)*T1
370 REM
380 IF ABS(V1)<S
                THEN 430
390 PRINT
     "BA･A･A･NG"
400 PRINT
     "ケ･ツメン ニ ケ･キトツ"
410 PRINT
     "ノリクミイン ゼ･ンイン
                シホ･ウ"
420 GOTO 450
430 PRINT
     "ケ･ツメン ニ フ･シ･
                チャクリク"
440 PRINT "オメテ･トウ"
450 REM
460 PRINT "チャクリク シ･カン
                :";T+T1
470 PRINT "チャクリク ソクト･
                :";V1
480 PRINT "ネンリョウ アマリ
                :";F
490 STOP

800 REM (グラフを書く)
810 PRINT "T=";T;"V="
          ;V;"H=";H;"F=";F
820 PRINT "I";TAB
            (H/9+1);"+"
830 RETURN
```

```
 10 REM ツキ チャクリク
                ケ･ーム
 20 PRINT
 30 PRINT "ツキ チャクリク
                ケ･ーム"
 40 PRINT
 50 REM イニシャライス･
 60 LET T=0
 70 LET V=40
 80 LET F=150
 90 LET H=200
100 LET S=2
110 REM (入力)
120 GOSUB 800
130 INPUT W
140 IF W*(W-30)<=0
                THEN 170
150 PRINT "フンシャリョウ ハ
      0カラ $30マテ･"
160 GOTO 130
170 IF W>F THEN W=F
            (計算部分, 1)
180 LET V1=V-0.5*
                W+2
190 LET F=F-W
200 LET H=H-0.5*
                (V+V1)
210 IF H<=0
                THEN 300
220 LET T=T+1
230 LET V=V1
240 IF F>0 THEN 110
250 IF W=0 THEN 280
260 PRINT "ネンリョウ キ･レ"
270 LET W=0
280 GOSUB 800
290 GOTO 180

300 REM (計算部分, 2)
310 LET H=H+0.5*
                (V+V1)
```

図3－49 月面着陸のプログラム

```
ツキチャクリク ケ゛ーム

T=  0  V=  40  H=  200  F=  150
I                 +
?
```

図3−50　ゲームの開始

```
I              +
I           +
I         +
I         +
I       +
I      +
I    +
I    +
I  +
```

図3−51　月面と宇宙船

```
T=0  V=  40  H=  200  F=  150
I                  +
?80
フンシャリョウ ハ 0 カラ 30 マテ゛
?10
T=1  V=  37  H=  176,500  F=140
I              +
?
```

図3−52　ゲームの進行

と出てきます。

入力した噴射量をまかなえるだけの燃料が残っている場合は、次のステップに移って、新しい宇宙船の状態と、グラフが出力されます（135頁・図3—52）。宇宙船の状態に対応しながら噴射量を入力していくというステップを何回か繰返して、無事に月面に着陸することができればよいのです。

着陸の速度

月面に到達した時の速度は、毎秒2キロメートル以上、2キロメートル以下でなければなりません。この速度範囲外の速度で月面に到達した場合、ショックが強すぎて乗組員が死んでしまいます。

無事に着陸できる速度の範囲は、2以上から2以下ですが、その値は100行の文で定められています。この値を大きくとれば、このゲームはやさしくなり、小さくすればむずかしくなります。

このゲームになれるにつれて、この値を小さくしてください。また、70～90行の速度、燃料、高さの初期値を変えることによっても、ゲームをむずかしくしたり、やさしくしたりなど変化させることができます。

プログラムの改良

図3—49のプログラムでも、かなり楽しく遊ぶことができますが、一度、無事に着陸する方法を覚えてしまうと、おもしろみが減ってきます。理由は、このプログラムが定められた式にのっとって行われているにすぎず、突発的な出来事が入ってくる余地がないからです。

この点を改善するために、今度は、乱数を使ってプログラムを作ってみました。

現実的にはあまり考えられないことですが、このプログラムでは、宇宙船が月面に到達する途中で、いくつかの流星群に出合います。そして、場合によっては、いくつかの流星と衝突して、燃料を失ってしまうかもしれません。

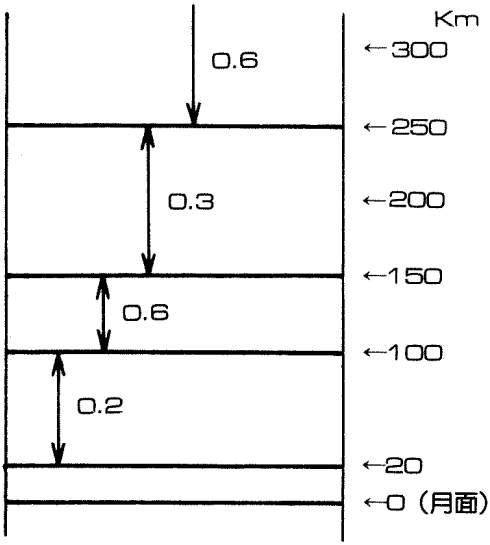


図３－53　流星群に出会う確率

この2つの出来事が出現するかどうかを、組み込み関数ＲＮＤを二度使って決めることにします。決め方は、図３―53、図３―54に示した通りです。

流星群に出合う確率は、高度によって異なります。

250 km以上と100 km～150 kmでは60％の確率で流星群に出合います。また、150 km～250 kmでは30％、

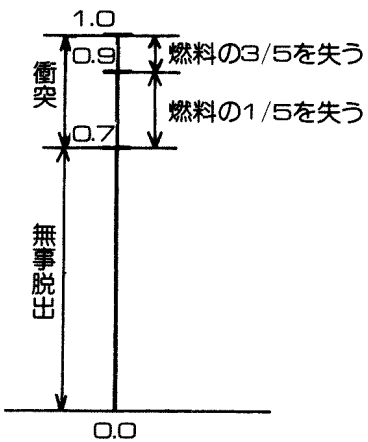


図３－54　流星群と出合った結果

20 km～100 kmでは20％です。

流星群に出合うと、「キケン」と表示されますが、必ずしも、宇宙船が流星とぶつかるとは限りません。

図3－54に示されるように70％までは無事に脱出できます。しかし残りの30％の場合は、流星に衝突し燃料を失ってしまいます。

以上のことを考慮に入れて作ったプログラムが図3－55です。これは、図3－49のプログラムの800行以下とおき代えるだけでよいのです。

```
800 REM (流星群)
810 LET S$=" "
820 IF H>=250 THEN R1=0.6
830 IF (H-150)*(H-250)<=0 THEN R1=0.3
840 IF (H-100)*(H-150)<=0 THEN R1=0.6
850 IF (H-20)*(H-100)<=0 THEN R1=0.2

860 IF RND(1)>R1 THEN 960
870 PRINT "キケン・・・リュウセイ ｸﾞﾝ アリ !!!"
880 LET R2=RND(1)

890 IF R2<0.9 THEN 920
900 LET F=2/5*F
910 GOTO 940

920 IF R2<0.7 THEN 960
930 LET F=4/5*F

940 LET S$="*"
950 REM
960 PRINT "T=";T;"H=";H;"V=";V;"F=";F
970 PRINT
    "I";TAB(H/9+1);S$;TAB(H/9+2);"="
980 IF S$="*" THEN PRINT "リュウセイ ニ ｹﾞｷﾄﾂ"
990 RETURN
```

図3－55　改良したサブルーチン

ライフ・ゲーム

マイコンのゲームとしてよく知られているものの1つに、ライフ・ゲームがあります。

ライフとは英語で「生命」という意味を持っています。このゲームは、生命が生まれ、そして死んでいく生物社会の状況をゲーム化したものです。

このゲームでは、1つ1つの生物（何かの符号で表わされます）は、碁盤のマス目のようなところに住んでいます。

そして、ある1つの生物の周囲のマス目にいる生物（これも符号で表わされています）の数によって、生存したり、死滅したり、また誕生したりします。

ゲームのルール

具体的なルールは次の3つしかありません。

① **生存するための条件**

ある生物の周囲に、ほかの生物が2つか3つあれば、その生物は生き残る。

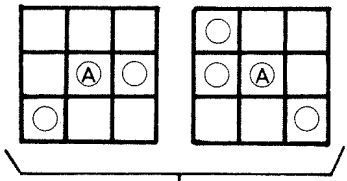
② **死滅するための条件**

ある生物の周囲に、ほかの生物が1つ以下しかないか、また4つ以上あれば、その生物は死ぬ。

③ **誕生の条件**

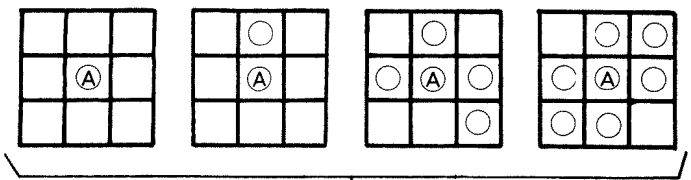
生物のいないマス目の周囲に3つ生物がいれば、そのマス目に新しく生物が誕生する。

この3つのルールは、生物が生きていくための条件、つまり孤独では生きていけないし、また、生物がたくさんいすぎても息苦しくなってしまい、そして死んでしまう、ということをよ



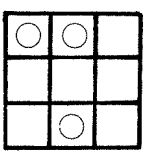
1．生存のための条件

生物Ⓐのまわりに
生物○が２つか３
つあればⒶは次の
世代まで生き残る



2．死滅のための条件

生物Ⓐのまわりに
生物○が１つ以下
か４つ以上あれば
Ａは次の世代まで
で死滅する



3．誕生のための条件

生物のいないマス目
のまわりに生物が３
つあれば、そこに生
物が誕生する

図３－56　ゲームのルール

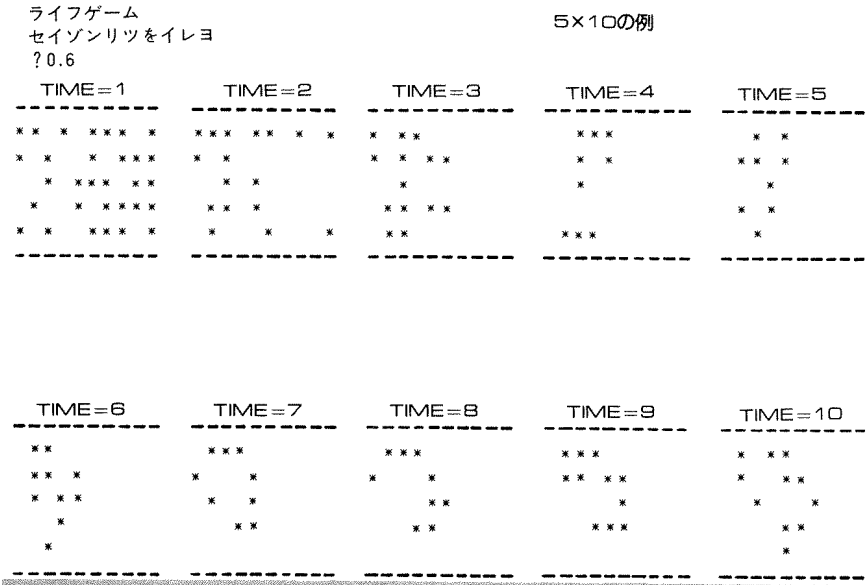


図3－57　ライフゲームの実行例

く表わしています(図3―56)。

このルールを、マス目に書き込まれた生物社会に適用するのですが、このときの約束としてこのルールは、ある世代の生物の集まりすべてに同時に適用し、それによって、次の世代を作っていかなければなりません。この注意は、あとでプログラムを作るときに必要となるので、覚えておいてください。

生物の社会（マス目のそこここに書き込まれた符号で示されています）に、この3つのルールにしたがって次々に操作をくりかえすと、生物の集まりは、色々な形に変化していきます。

ライフ・ゲームは、この形の変化をみて楽しむゲームで、私たちが、特別にマイコンに指示する必要はありません。最初の生物社会の状態(初期値)を決めれば、あとはマイコンが次々と異なった画面を示してくれます(図3―57)。

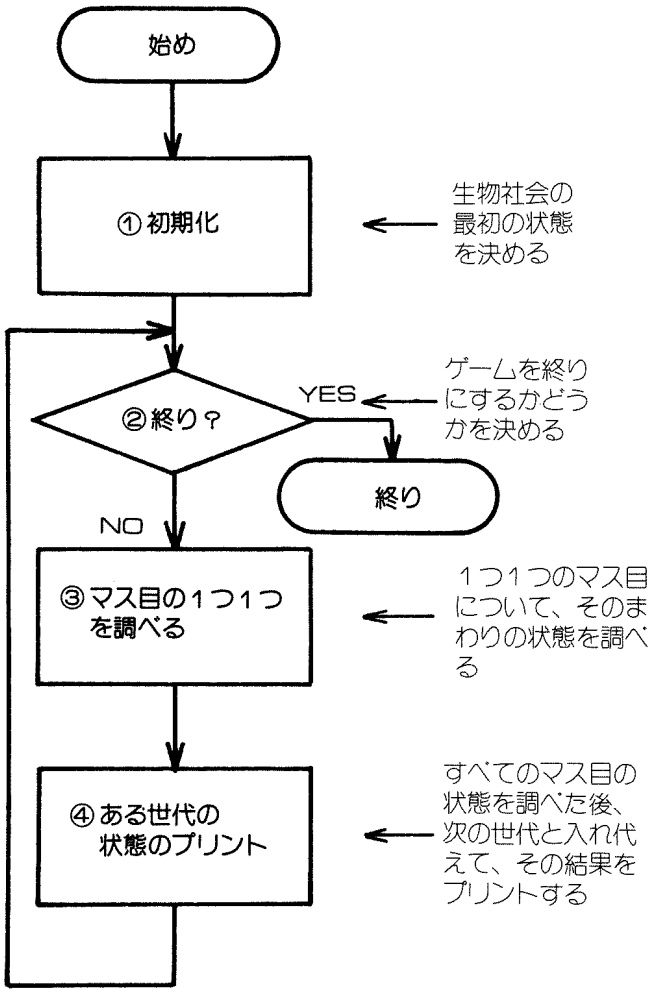


図3－58　ライフ・ゲームのフローチャート

ライフ・ゲームのプログラム

ライフ・ゲームのプログラム自体は、それほどむずかしくはありません。マス目としては、ベーシックの配列（DIM）を使います。ゲームという目的からは、そのマス目の数が多ければ多いほどよいのですが、画面の大きさ（ふつうのテレビの画面では100×100の大きさのマスはつくれません）や、プログラムの大きさ（やはりDIM S(100, 100)のような大きい配列は記憶容量との関係で作れないこともあります。）に制約されることが多いようです。ここでは、マス目の大きさは15×15としましたが、みなさんのマイコンやTVディスプレイの画面の大きさに応じて、プログラムを直して使ってください。大きければ大きいほどよいのです。

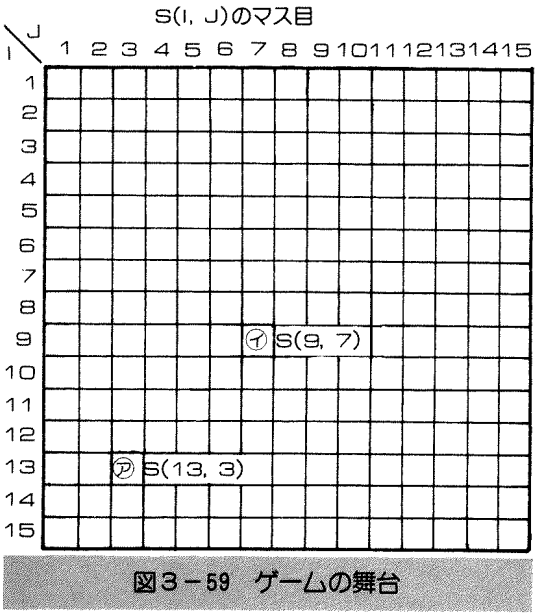


図3－59　ゲームの舞台

ライフ・ゲームのプログラムのためのフローチャートは、図3—58に示した通りで、それほどむずかしくはありません。

ゲームの舞台としてのマス目は、配列S(15, 15)で現わします。したがって図3—59のように、15×15で225個のマス目を考えているわけです。そしてその1つ1つについては生物がいる

S(I, J) の値

		次の世代	
		生	死
今の世代	生	2	−2
	死	−1	1

S(I, J) の値が
2のとき　今の世代でも次の世代でも生き残る。(生存)
−2のとき　今の世代では生きているが、次の世代で死ぬ。(死滅)
1のとき　今の世代でも次の世代でも死んだまま。
−1のとき　今の世代では死んでいるが、次の世代で生まれる。(誕生)

図3−60　生物の生死をプログラムで示す方法

かいないかを調べ、また次の世代で生まれるか、死ぬかを考えなければならないのです。

配列の見方はいままでと同じで、たとえば、図3−59の㋐は左から3列目、上から13行目にあるのでS(13、3)と表わされ、㋑は、左から7列目、上から9行目にあるので、S(9、7)と表わされます。

マス目を変数で表わす

プログラミングを始める前に、いくつかの約束を決めておかねばなりません。マイコンでは「ある生物が生きている」という情報を、そのままの形で記憶する力はありませんから、生きているかいないかは、私たちがマイコンにわかる言葉にして決めなくてはならないわけです。

ここでは、マス目I行J列　(S(I, J))にいる生物が生きている場合は2を、死んでいる場合は1を、配列変数S(I, J)にあたえることに

します。したがって、あるときに、S(13, 3)が1ならば、13行3列（図3—59の㋐点）には生物がいないのです。また、S(13, 3)が2ならば、そこに生物がいると決めるわけです。

ただし、このゲームでは、ある時点で生物が生きているか死んでいるかの判断だけではなく、次の世代に代ったときにどうなるかも知らなければなりません。ここでは、次の世代に代ったときに状態が変る（生から死へ、また逆に死から生へ）としたら、その値は変えずに符号だけ変えることにします。

つまり、I行J列目の生物がどういう状態かをみるためにはS(I, J)の値を調べればよいのですが、その値が2ならば、今の世代で生きていて次の世代にも生きのびることを示し、また、その値が−2ならば今の世代では生きているが、次の世代では死んでしまうことを示しているこ

とにするわけです。同様にS(I, J)の値が1ならば、今の世代でも次の世代でもI行J列の地点には生物は存在しません。1ならば、今の世代では生物は存在しないけれども、次の世代で新しく誕生するのです。この約束をまとめたものは図3—60に示しておきました。この約束を使ってプログラミングをすればよいのです。

プログラム作成上の注意

ライフゲームのプログラムの中心部分をしめるのは、1つ1つのマス目について、その周囲の状態を調べていくところです（図3—58のフローチャート参照）。146頁の図3—61のように、S(I, J)の周囲には、8つの点があり、それぞれに生物がいたり、いなかったりします。したがって、S(I, J)の周囲の8つの点の値の合計によって、S(I, J)の次の状態が決定されるのです。

S(I−1、J−1)	S(I−1、J)	S(I−1、J+1)
S(I、J−1)	S(I、J)	S(I、J+1)
S(I+1、J−1)	S(I+1、J)	S(I+1、J+1)

図3−61　S(I、J)の周囲

しかし、ここで注意するべきことが2つあります。1つは、配列の上限と下限です。（このプログラムでは、わかりやすくするために、配列の下限は1であると考えています。）

IやJは、1から15まで変わりますから、S(I−1,J−1)やS(I+1,J+1)などでは、S(0、0)、S(16、16)というようになってしまい、配列の上限と下限をとび出してしまうことになるのです。したがって、そのためのチェックをしなければならないのです。

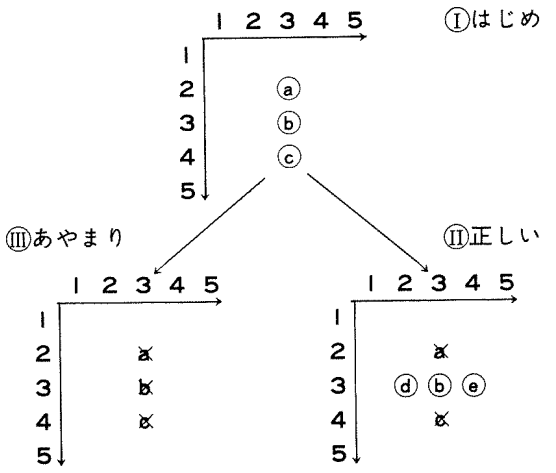


図3−62　奇妙なことが起る

また、先ほどのライフ・ゲームのルール説明のときに注意として書いたことにも気をつける必要があります。その注意とは、ライフ・ゲームの3つのルールは、ある世代の生物の集まりすべてに同時に適用しなければならないということでした。この注意を守らないと、少し奇妙なことが起ります。たとえば図3—62のように①から出発したとします。本来はⓐとⓒがルール2（図3—56参照）によって消え、ⓓとⓔがルール3によって誕生するはずでした。しかし、注意をまもらず、1つ1つについて調べてすぐに状態を書き換えてしまうと、図3—62のⅢのようにS(2，3)がルール2によって消えると、そこには、何もなくなってしまうことになるので、S(3，3)もルール2によって消え、結局、すべてが死滅してしまうわけです。

プログラムの作成

以上のことに注意して、プログラムを書くと149頁・図3—63のようになります。

図3—58のフローチャートと対照してみると、初めにDIMでS(15，15)の配列をとっています。このプログラムでは15×15にしましたが、みなさんのマイコンにあわせて大きさをかえるときには30・40行のX1、Y1も変えてください。

フローチャートの①初期化の部分が50～150行です。ここでは、生存率Rを適当に入力し（ただし0以上1以下）それに応じて、マイコンが最初の生物の分布のようすを決めてくれます。

たとえば、生存率が0.3ならば全体のほぼ30％（15×15のマス目の30％、すなわち約70個のマス目）に生物が生きていることになります。

ここから出発して170行のINPUT文で何かが入力されるごとに新しい画面に移っていきます。このINPUT文では、何を入力してもよ

いのですが、999が入力されたときにかぎってストップします。このとき、ゲームは終ります。

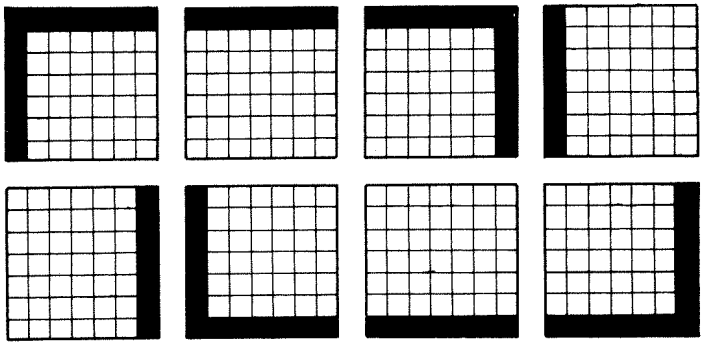
190～530行までは、マス目の1つ1つについて調べているところです。ここで使われている関数ABSは、絶対値を求める関数です(図3―64)。

540～600行では、それぞれのマス目の周囲を調べた結果、図3―60に示される値をS(I, J)に代入しています。

2重のFOR、NEXT文によってすべてのマス目について、調べられたあと、630～750行で結果の書き出しがおこなわれます。出力結果は図3―57に示されているようになります。

```
ABS(−3)=3
ABS(3)=3
ABS(3.5)=3.5
ABS(−10)=10
```

図3－64　関数ABSの使い方



S(I, J)が左図の黒い色のところにあるときには注意が必要

S(I、J)が黒い部分にきたときには、C1～C8までのどれかが配列Sの上限が下限をこえてしまうので注意して例外処理を考える必要があります

黒い箇所が周辺にきたときの扱い方

図3－63　ライフゲームのプログラム

```
 10 REM
      (ライフゲームのプログラム)
 20 DIM S(15, 15)
 30 LET X1=15
 40 LET Y1=15
 42 LET T=0
 44 PRINT
      "***ライフゲーム***"
 46 PRINT

 50 REM (イニシャライズ)
 60 PRINT
      "セイソ・ンリツヲ　イレヨ"
 70 INPUT R
 80 RANDOMIZE
 90 FOR I=1 TO X1
100 FOR J=1 TO Y1
110 LET S(I, J)=1
120 IF RND>R THEN 140
130 LET S(I, J)=2
140 NEXT J
150 NEXT I
160 GOTO 630
165 REM

170 INPUT E
180 IF E=999 THEN 999
190 REM (S(I, J)の
          まわりの状態を調べる)
200 FOR I=1 TO X1
210 FOR J=1 TO Y1

220 IF(I-1)*(J-1)=0
                THEN 250
230 LET C1=
          ABS(S(I-1, J-1))
240 GOTO 260
250 LET C1=1

260 IF I=1 THEN 290
270 LET C2=
            ABS(S(I-1, J))
280 GOTO 300
290 LET C2=1

300 IF(I-1)*(J-Y1)=0
                THEN 330
310 LET C3=
          ABS(S(I-1, J+1))
320 GOTO 340
330 LET C3=1

340 IF J=1 THEN 370
350 LET C4=
            ABS(S(I, J-1))
360 GOTO 380
370 LET C4=1

380 IF J=Y1 THEN 410
390 LET C5=
            ABS(S(I, J+1))
400 GOTO 420
410 LET C5=1

420 IF(I-X1)*(J-1)=0
                THEN 450
430 LET C6=
          ABS(S(I+1, J-1))
```

```
634 PRINT
636 PRINT "    TIME=";T

640 GOSUB 800
650 FOR I=1 TO X1
660 FOR J=1 TO Y1
670 IF S(I, J)=-2
              THEN S(I, J)=1
680 IF S(I, J)=-1
              THEN S(I, J)=2

690 IF S(I, J)=1
    THEN P$=" "(死んでいる)
700 IF S(I, J)=2
    THEN P$="*"(生きている)

710 PRINT TAB(J-1);P$;
720 NEXT J
730 PRINT
740 NEXT I
750 GOSUB 800
760 GOTO 170

800 REM (ワクヲ カク)
810 FOR K=1 TO Y1
820 PRINT TAB(K-1);"-";
830 NEXT K
840 PRINT
850 RETURN

999 STOP
```

```
440 GOTO 460
450 LET C6=1

460 IF I=X1 THEN 490
470 LET C7=
            ABS(S(I+1, J))
480 GOTO 500
490 LET C7=1

500 IF(I-X1)*(J-Y1)=0
                THEN 530
510 LET C8=
          ABS(S(I+1, J+1))
520 GOTO 540
530 LET C8=1

540 REM
550 LET D=C1+C2+C3+C4
      +C5+C6+C7+C8-8
560 IF S(I, J)=2
                THEN 590
570 IF D=3 THEN
          S(I, J)=-1→誕生
580 GOTO 610

590 IF(D-2)*(D-3)=0
          THEN 610→生存
600 LET S(I, J)=-2    →死滅

610 NEXT J
620 NEXT I

630 REM (カキダシ)
632 LET T=T+1
```

スタートレック

スタートレック（ＳＴＡＲ・ＴＲＥＫ）は、「星間旅行」というほどの意味で、何年か前にアメリカのテレビ局が放映したＳＦ物語の題名です。

そのテレビ番組を、コンピュータ・ゲームにしたものが、スタートレック・ゲームですが、最初は、大型コンピュータやミニコンのデモ・プログラム（宣伝のためのプログラム）として作られました。マイコン・ゲームとして親しまれるようになったのは、しばらくあとになってのことです。ところが、マイコンで楽しめるようになったスタートレックは、爆発的な人気を呼んで、マイコン・ゲームというと、スタートレックがその代表といわれるようになりました。マイコン出現以来、数多くのゲームが作られましたが、スタートレックは、いまだに王座を守っています。スタートレックのプログラムは、ベーシックで書かれているのが普通です。

スタートレックのプログラムは、短いもので100行弱、長いものでは1000行以上になります。内容もさまざまです。1000行以上のプログラムは、マイコン用ではなく、大型コンピュータ用に作られたものが多いようです。

ゲームの内容

主人公は、宇宙船エンタープライズ号です。テレビ番組では、カークという名前の人がエンタープライズ号の船長ですが、マイコン・ゲームでは私たちが船長です。

エンタープライズ号の任務は、宿敵ロムラン星人の繰つる宇宙船クリンゴンを、宇宙の果ま

で追い求め撃滅することです。クリンゴンは、1隻だけではなく何隻もいて、広大な宇宙のあちこちで私たちを待ち受けています。

小宇宙

宇宙（ギャラクシー）は、いくつかの小宇宙（クォドラント）に分れています。小宇宙は8×8の大きさで、その中にいくつかの星（＊）と、クリンゴン（＝）、そして私たちのエンタープライズ号（＃）が配置されています（図3－65）。大きなプログラムでは、私たちが今、宇宙全体の中でどの小宇宙にいるかも表示できるのですが、このプログラムでは、宇宙全体の中での小宇宙の位置については考えません。

```
  1 2 3 4 5 6 7 8
1     ＊       1
2   ＝     ＝   2
3 ＊         ＊ 3
4       ＝     4
5 ＝ ＊       ＊ 5
6             　6
7 ＊   ＊   ＃ 　7
8             　8
  1 2 3 4 5 6 7 8
```

図3－65　クォドラントの表示

エンタープライズ号（＃）

エンタープライズ号は、8×8の大きさの小宇宙を動き回って、クリンゴンを攻撃します。エンタープライズ号の動きを指示するのは、船長の私たちです。エンタープライズ号の機能は、

① 小宇宙内を移動すること。
② クリンゴンに対して波動砲（フェーザ）攻撃を行うこと。
③ エネルギーを全部使わないうちに、別の小宇宙に移ること。

です。ゲームの最初にエンタープライズ号に与えられたエネルギーは、①と②によってだんだん少くなってきます。③を行うとエネルギーは自動的に補充されます。プログラムでは多くエネルギーの補充をスターベース（味方の宇宙基地）に立寄って行うのですが、このプログラムでは、省略しました。

クリンゴン（三）

小宇宙には、5隻以下のクリンゴンがいます。クリンゴンがエンタープライズ号を攻撃することはありませんが、エンタープライズ号が近づくと逃げ回ります。エンタープライズ号の波動砲は、距離が遠くなると、ほとんど効果を持ちません。それは、クリンゴンがシールド（攻撃から身を守る装置）をしているからです。このシールドは、クリンゴンによって強さが違います。シールドの弱いクリンゴンは、遠くからの波動砲攻撃によっても簡単に破壊されてしまいます。ただ、どのクリンゴンのシールドが弱いのかを知ることはできません。1番確実なのはクリンゴンにできるだけ接近して十分なエネルギーを使った波動砲攻撃をすることです。

ゲームの遊び方

プログラムの実行を始めると、まず154頁の図3—66のような表示が画面に出てきます。小宇宙内のエンタープライズ号、クリンゴン、そして、星の配置が図示されます。クリンゴンが、バラバラに、しかもエンタープライズ号から遠く配置されている時は、全部を撃滅するのがむずかしくなります。

最初の配置は、プログラムの中で乱数を使って決められますから手のほどこしようがありません。

メッセージ

小宇宙の地図の横には、その時点でのエンタープライズ号の位置、エネルギー、そしてクリンゴンの数が表示されます。エンタープライズ号の位置とクリンゴンの数は、小宇宙の地図からもわかりますが、エネルギーは、ほかで表示されることはありません。このエネルギーが0になってしまうと、エンタープライズ号は動けなくなってしまい、ゲームは、私たちの負けになります。なおエンタープライズ号の位置が、

　イチ　：48

となっていれば、小宇宙では、上から4、左から8の地点を示します。

指令（コマンド）

小宇宙地図の下に、

　シレイ　ヲ　ト゛ウソ゛

と出るのは、エネルギーがまだ残っているとき

```
  12345678
1    =        1    エンターフ゜ライス゛
2  *          2
3       *     3    イチ        :  55
4             4    エネルキ゛ー:  200
5  =   *#    =5    クリンコ゛ン:  3
6             6
7  *          7
8         *   8
  12345678

シレイ  ヲ  ト゛ウソ゛
?1
?67
```

図3－66　ゲームの開始

だけです。ここで、船長の私たちの指令を伝えます。「?」のあとに数字で指令を入力します。

指令「1」

コマンド1は、エンタープライズ号のエンジンを制御して、小宇宙の任意の場所に移動するためのものです(図3―67)。「1」を入力すると、続いて「?」が出てきますから、行きたい地点を入力します。たとえば、上から6、左から7の地点に行きたいときは「67」を入力すればよいのです。星のある地点を行き先として入力すると、

ﾎｼ ﾆ ｼｮｳﾄﾂ!!

というメッセージが出て、エンタープライズ号は破壊され、ゲームは私たちの負けです。クリンゴンにぶつかると、

ｸﾘﾝｺﾞﾝ ﾆ ｹﾞｷﾄﾂ!!

というメッセージが出ます。クリンゴンも、エ

```
   12345678
  1        1    ｴﾝﾀｰﾌﾟﾗｲｽﾞ
  2  *  =  2
  3     *  3    ｲﾁ      :  67
  4  =    =4    ｴﾈﾙｷﾞｰ  :  190
  5    *   5    ｸﾘﾝｺﾞﾝ  :  3
  6       #6
  7  *     7
  8      * 8
   12345678

ｼﾚｲ ｦ ﾄﾞｳｿﾞ
?2
?40
ｸﾘﾝｺﾞﾝ 47 ﾊﾞｸﾊ !!
```

図3―67　図3―66で入力したコマンド「1」の結果

```
  12345678
1          1   エンターフ゜ライス゛
2  *  =    2
3     *    3   イチ        :  47
4          4   エネルキ゛ー:  150
5    =*    5   クリンコ゛ン:  2
6       #  6
7  *       7
8      *   8
  12345678

シレイ  ヲ  ト゛ウソ゛
?1
?44
```

図3－68　図3－67で入力したコマンド「2」の結果

ンタープライズ号も同時に破壊されてしまいます。これは「特攻」ですが、エンタープライズ号は1隻しかないのですから、やはり私たちの負けです。

移動によって、エネルギーは減少します。あまり遠くに1回で移動すると、エネルギーの消費が激しくなりますから、注意してください。

指令「2」

クリンゴンに十分近づいたところで、コマンド2の波動砲を使います。「2」を入力すると、「?」がもう1つ出ますから、波動砲に投入するエネルギーを入力します(図3—68)。投入するエネルギーが大きければ大きいほどクリンゴンのシールドを破る可能性も大きくなります。

波動砲には方向がありません。その効果は、全小宇宙に拡がります。ですから、クリンゴンによってシールドの強さが違うと近いクリンゴン

は無事で、遠いクリンゴンが破壊されるということも起りえるでしょう。

指令「3」

エネルギーの残っているうちに小宇宙内のクリンゴンを全滅できれば、私たちの勝ちです。クリンゴンを求めて次の小宇宙に旅立つためにコマンド3を使います。「3」を入力するとエネルギーが補充され、新しい小宇宙の中にいるエンタープライズ号が現われてきます。

エネルギーの使い方をよく考えなければ、クリンゴンを全滅させることはできません。クリンゴンも逃げ回りますから、エンタープライズ号の移動地点にも頭を使う必要があります。

クリンゴンは逃げるときには必ず、上下左右に1単位ずつ動きます。エンタープライズ号と違って、遠いところにジャンプすることはありません。その点を利用すれば上手にクリンゴンを追いつめられるでしょう。

指令「4」

コマンド4は、ゲームを終らせるための指令です。

最初に説明したように「スタートレック」のプログラムは無数にあります。共通しているのは、エンタープライズ号がクリンゴンを攻撃するという点だけだとさえいえます。このプログラムは、無数のプログラムの中では比較的短いもので、エンタープライズ号の機能もそれほど豊富ではありません。大型コンピュータ用のスタートレックでは覚えられないくらいたくさんのコマンドがあって、遊び方を理解するのに苦労するくらいです。

プログラムの説明

プログラムの内容を、簡単に説明します。

100～330行　小宇宙G(8, 8)に適当にエンタープライズ号、クリンゴン、星を配置する。

340～660行　小宇宙の様子を画面に表示する。

670～770行　適当なメッセージを出力しコマンドを入力する。

780～940行　コマンドが「1」のときの処理です。

950～1110行　コマンドが「2」のときの処理です。

1120～1300行　クリンゴンを動かし新しい小宇宙地図を作ります。

1310～1410行　最初の配置を作るために使うサブルーチンです。

1420～1480行　小宇宙地図を表示する為に使うサブルーチンです。

図3－69　スタートレックのプログラム

```
100 REM STAR TREK
                (APOCRYPHA)
110 DIM G(8, 8), N$(8)
120 FOR I=1 TO 8
130 READ N$(I)
140 NEXT I
150 DATA "1", "2",
                "3", "4"
160 DATA "5", "6",
                "7", "8"
170 FOR I=1 TO 8
180 FOR J=1 TO 8
190 LET G(I, J)=0
200 NEXT J
210 NEXT I
220 LET E=200
230 LET S=INT
          (RND(1)*9)+1
240 LET K=INT
          (RND(1)*5)+1
250 LET J=1
260 LET R=1
270 GOSUB 1310
280 LET J=2
290 LET R=S
300 GOSUB 1310
310 LET J=3
320 LET R=K
330 GOSUB 1310
340 GOSUB 1420
350 FOR I=1 TO 8
360 PRINT " ";N$(I);
```

```
690 IF E>0
                    THEN 720
700 PRINT
        〝エネルキ゛ー フソク〟
710 STOP
720 PRINT
        〝シレイ ヲ ト゛ウソ゛〟
730 INPUT M
740 IF M=1
                    THEN 780
750 IF M=2
                    THEN 950
760 IF M=3
                    THEN 170
770 STOP
780 INPUT Q
790 LET X=
                    INT(Q/10)
800 LET Y=Q-10*X
810 IF G(X, Y)<0
                    THEN 840
820 IF G(X, Y)=2
                    THEN 860
830 GOTO 880
840 PRINT 〝クリンコ゛ン
        ニ ケ゛キトツ !!〟
850 STOP
860 PRINT
      〝ホシ ニ ショウトツ !!〟
870 STOP
880 LET I=INT(P/10)

890 LET J=P-10*I
900 LET D=(X-I)*(X-I)
            +(Y-J)*(Y-J)
910 LET G(I, J)=0
920 LET G(X, Y)=1
930 LET E=E-2*D
940 GOTO 1120
950 INPUT B
370 FOR J=1 TO 8
380 LET C=G(I, J)
390 IF C<0
            THEN 440
400 IF C=1 THEN 480
410 IF C=2
                THEN 460
420 PRINT 〝 〟;
430 GOTO 500
440 PRINT 〝=〟;
450 GOTO 500
460 PRINT 〝*〟;
470 GOTO 500
480 PRINT 〝#〟;
490 LET P=10*I+J
500 NEXT J
510 IF I=1 GOTO 570
520 IF I=3 GOTO 590
530 IF I=4 GOTO 610
540 IF I=5 GOTO 630
550 PRINT N$(I)
560 GOTO 640
570 PRINT N$(I);〟
        エンターフ゛・ライス゛〟
580 GOTO 640
590 PRINT N$(I);〟
            イチ      :〟;P
600 GOTO 640
610 PRINT N$(I);〟
          エネルキ゛ー:〟;E
620 GOTO 640
630 PRINT N$(I);〟
            クリンコ゛ン:〟;K
640 NEXT I
650 GOSUB 1420
660 PRINT
670 IF K>0
                THEN 690
680 PRINT
      〝クリンコ゛ン セ゛ンメツ〟
```

```
1220 IF X>8
                THEN 1170
1230 IF Y<1
                THEN 1170
1240 IF Y>8
                THEN 1170
1250 IF G(X, Y)<>0
                THEN 1170
1260 LET G(X, Y)=
                        G(I, J)
1270 LET G(I, J)=0
1280 NEXT J
1290 NEXT I
1300 GOTO 340
1310 REM FIRST MAP
1320 FOR I=1 TO R
1330 LET X=INT
            (RND(1)*8)+1
1340 LET Y=INT
            (RND(1)*8)+1
1350 IF G(X, Y)<>0
                THEN 1330
1360 IF J=3 THEN 1390
1370 LET G(X, Y)=J
1380 GOTO 1400
1390 LET G(X, Y)=
        INT(-RND(1)*10)
1400 NEXT I
1410 RETURN
1420 REM SCALE
1430 PRINT "    ";
1440 FOR I=1 TO 8
1450 PRINT N$(I);
1460 NEXT I
1470 PRINT
1480 RETURN
1490 END
 960 LET X=INT(P/10)
 970 LET Y=P-10*X
 980 LET H=0
 982 FOR I=1 TO 8
 984 FOR J=1 TO 8
 990 IF G(I, J)>=0
                THEN 1070
1000 LET D=(X-I)*(X-I)
            +(Y-J)*(Y-J)
1010 IF G(I, J)<-B/D
                THEN 1070
1020 LET H=H+1
1030 LET K=K-1
1040 LET P=10*I+J
1050 PRINT "ｸﾘﾝｺﾝ"
                ;P;"ﾊﾞｸﾊ !!"
1060 LET G(I, J)=0
1070 NEXT J
1080 NEXT I
1090 IF H>0
                THEN 1110
1100 PRINT "ﾄﾞｳﾎｳ
            ｴﾈﾙｷﾞｰ ﾌｿｸﾞ"
1110 LET E=E-B
1120 REM NEW MAP
1130 LET H=0
1140 FOR I=1 TO 8
1150 FOR J=1 TO 8
1160 IF G(I, J)>=0
                THEN 1280
1170 LET X=I+INT
            (RND(1)*3)-1
1180 LET Y=J+INT
            (RND(1)*3)-1
1190 LET H=H+1
1200 IF H>100
                THEN 340
1210 IF X<1
                THEN 1170
```

囚人のジレンマ

ある国で、2人組の銀行強盗がつかまりました。証拠がなかったので、何とか2人に自白させたいのです。2人は、別々の部屋に、監禁されていて、話し合うことはできません。

その国の法律では、2人の囚人のうち、どちらかが自白すると、その囚人は、すぐ釈放され、相棒は、その代りに懲役10年の刑を受けることになっています。そうかといって、2人とも同時に自白してしまうと、2人とも釈放されるのではなく、それぞれ懲役5年にされます。2人とも自白しないと、当局は困って、2人を、懲役1年の刑にしてしまいます。

2人の囚人を、A、Bとしましょう。囚人Aは、次のように考えます。

囚人の自問

「Bのやつが黙っていてくれるのなら、おれも黙っていよう。2人一緒に1年くらい刑務所で我慢してもいいじゃないか。おれが、先に自白しちまうと、あいつに10年も臭い飯を食わせることになるからな。あいつと組んで仕事を始めてから、かなり長いことになるし、1年たって、新しい計画をたててみるのも悪くない。でも、待てよ、Bのやつが、もう自白しちまったということはないかな。

もし、そうなら、こっちにも自白しなけりゃならないぞ。2人、一緒に自白すると、5年の刑になるはずだ。

1人で10年よりは、ずっとましじゃないか。それに、Bが黙っていたなら、おれは、自白してすぐ釈放だ。

でも、そうすると、Bを裏切ったことになる

だろうな。裏切りは、仲間うちでも御法度だ。釈放されても、仕事ができねえんじゃ、どうしようもない。Bも、おれも黙っていて、1年だけおつとめするのが、1番いいんだが。でも10年はいやだ。10年たって、出られて、年はいくつになるっていうんだ。ちきしょう、この仕事さえ、うまくいってりゃ、今頃、かっさらったお金で高飛びできてたんだ。あのとき、Bのやつが、へまをしなけりゃ、足がつくこともなかったんだ。吐いちまおうか。悪くても5年だ。うまくいきゃ、釈放じゃねえか。でも、……」

いくらか考えすぎのようですが、囚人Bも同じことを考えるとします。

このような状況は、まったくのジレンマなのです。あちらをたてれば、こちらがたたず、困ってしまう状態です（図3—70）。

（Bの行動）
Aの行動

Bの行動（Aの行動）	黙秘	自白
黙秘	懲役1年	釈放
自白	懲役10年	懲役5年

Aはどうなるか?
（Bはどうなるか?）

図3－70　囚人の行動と結果

マイコンとのジレンマ・ゲーム

このジレンマを、マイコンと私たちの間で演じてみましょう。

囚人のジレンマは、理論的には1回だけのゲームとして扱われます。繰返して行われる場合も「もっとも損害が少ない作戦」が理論的興味の中心です。でも私たちにとっては、このゲームを「面白く遊べること」が問題なのですから最適戦略は考えません。

ゲームの単純化

マイコンとのジレンマ・ゲームは、このジレンマ状況を何回も繰返して、戦略を互いに予想するようなものにします。そのためにジレンマを、もう少し単純化しておきます。自白や黙秘の代りに、作戦1、2と呼び、懲役何年というのではなく、得点が何点という条件にしましょう。そして、マイコンと私たちが、同時に作戦を出し合い、それぞれの得点を計算します。目標は、マイコンと私たちの得点が、ある一定の水準に達することです（164頁・図3—71）。

マイコンの戦略

マイコンがいつも同じ手でくるのでは面白くありませんし、まったくランダムな手でくるのもつまりません。ここではマイコンが私たちの作戦にしたがって自分の作戦を変化させていくことを考えます。マイコンが私たちの戦略を分析して、自分の戦略を決め、私たちがその裏をかくことができれば、私たちの勝ちです。いわば、マイコンとの知恵くらべをするわけです。なぜ、そのようなことができるのかはあとにして、ゲームの説明に入ります。

ゲームの遊び方

ルールはきわめて単純です(図3—72)。「?」が出てきたら、あなたの作戦（1か2）を入力してください。マイコン側の作戦が続いて出力されます。注意してほしいのは、あなたの入力をみてからマイコンが、自分の作戦を決めているのではないことです。このことは、プログラムの説明のところでわかるでしょう。

マイコン側の作戦と同時に、あなたとマイコンの得点が出力されます(166頁・図3—73)。この得点は累計得点で、ゲームの最初から1回ごとの得点を加えてきたものです。この得点をマイコン側も、あなたの側も、一定の得点（たとえば20点）になるようにマイコンを「だます」のが仕事です。実行してみるとわかりますが、マイコンは自分の得点を増やすことしか考えま

私たちの作戦

マイコンの作戦	1		2	
1	1	1	−2	2
2	2	−2	−1	−1
	マイコンの得点	私たちの得点	マイコンの得点	私たちの得点

図3−71　マイコンと私たちの得点表

```
シュウシ゛ン　ノ　シ゛レンマ

アナタ　モ　ワタシ　モ　サクセン　１，２　ノ
ト゛チラカ　ヲ　トル　コト　カ゛　テ゛キ　マス

アナタ　ノ　トクテン　　ワタシ　ノ　トクテン
　　１　　２　　　　１　　２
ワ　　　　　　ア
タ　１　　１　　２　　ナ　１　　１　　２
シ　２　−２　−１　　タ　２　−２　−１

サクセン　１　ハ　ヨワキ　ノ　サクセン
サクセン　２　ハ　ツヨキ　ノ　サクセン　テ゛ス

ワタシ　タチ　カ゛　ソレソ゛レ　２０　テン　ニ
ナレハ゛　ケ゛ーム　ハ　オワリ　テ゛ス
```

図３−72　ゲームの開始

せんから、あなたの作戦の立て方によっては、かなり難しい仕事になります。

マイコン側が強気に出てきたり、私たちの側の作戦を単純にしすぎてマイコンに読まれたりすると、「両者の得点を一定にする」という目標は達成できません。作戦２は、強気の作戦で、作戦１は、弱気の作戦です。マイコンは最初、強気で出るようにセットされています。あなたの側の作戦をいろいろ変えていくと、マイコン側も、それに対応して、作戦を変えていくはずです。

マイコンの学習

プログラムの説明に入る前に、マイコンの「学習」について考えてみましょう。私たちが一つの態度を取りつづけると、マイコンの側でもそれに合うように自分の態度を調整し、私たちが

```
?1
  1      アナタ    1      ワタシ    1
?2
  2      アナタ    0      ワタシ    0
?1
  2      アナタ   −2      ワタシ    2
?2
  2      アナタ   −3      ワタシ    1
?1
  1      アナタ   −2      ワタシ    2
?2
  1      アナタ    0      ワタシ    0
```

図3−73　ゲームの進行

態度を変えると、マイコンの方でも態度を変えるなら、マイコンは、私たちの態度を手がかりに「学習」しているわけです。

たとえば、ネコに、箱Pと箱Qをみせ、箱Pの側に行くとエサをやります。箱Qの方に行くとエサはやらず殴りつけます。そうするとネコは、だんだん箱Pの方にばかりいくようになるでしょう。

マイコンについても同じことを考えられないでしょうか。マイコンが「P」と出力するとエサをやるかわりに「1」と入力し、「Q」と出力すると殴りつけてもしかたがありませんから、「0」と入力します。マイコンは「1」が入力されると「今の出力はよかったんだ」と考え、「0」が入力されると「ダメだったんだ」と考えます。そして自分の態度（この場合、PとQのどちらかを出力するか）を変えるわけです（167

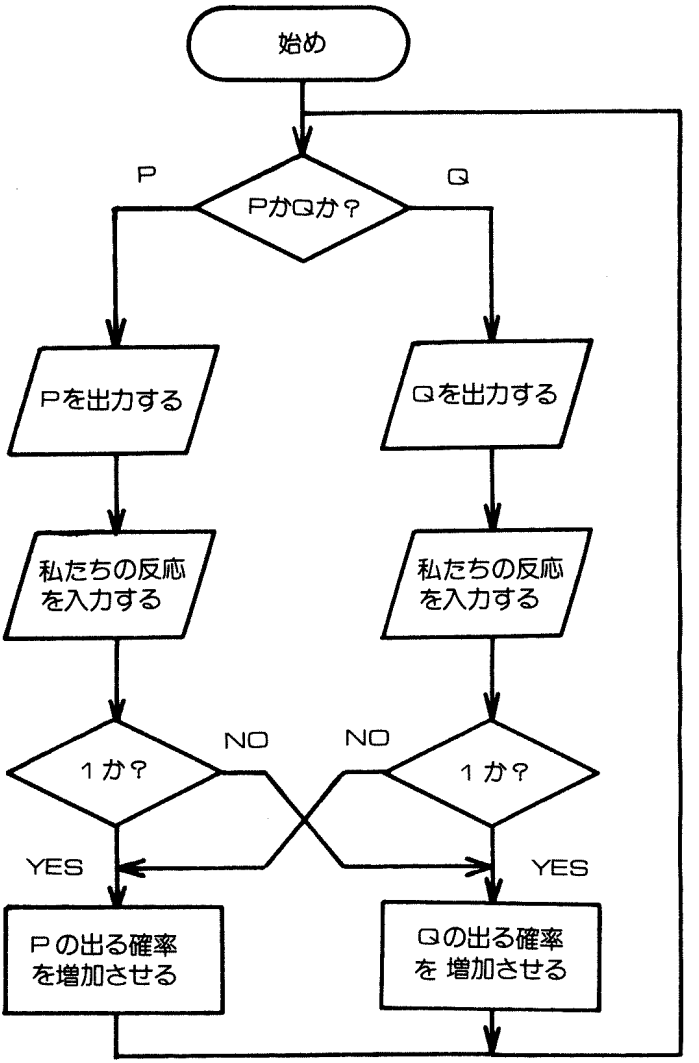


図3－74　マイコンの学習

```
100 REM LEARNING
110 PRINT "カ・クシュウ"
120 PRINT
130 LET A=0.2
140 LET P=0.5
150 LET C=RND (1)
160 IF C>P THEN 210
170 PRINT "      P"
180 INPUT D
190 IF D=0 THEN 240
200 GOTO 260
210 PRINT "      Q"
220 INPUT D
230 IF D=0 THEN 260
240 LET P=(1−A)*P
250 GOTO 150
260 LET P=(1−A)*P+A
270 GOTO 150
280 END
```

図3－75　マイコンの学習プログラム

頁・図3―74）。

このようなことをするプログラムを図3―75に示します。かえていくのは、Pを出力する確率です。Pを出力する確率が大きくなれば、Qを出力する確立が小さくなるのは当然です。

このプログラムの特徴は、私たちが途中で態度をかえたり（今までは、Pに対して1を入力していたが、これからはQに対して1を入力する）、入力を間違えたり（Pが正しいのに、0を入れてしまう）しても、それに対して、適当に対応してくれることです（図3―76・77）。

ジレンマ・ゲームのプログラム

マイコンの「学習」を、もっと複雑な条件で考えたのが、ジレンマ・ゲームのプログラムです（170頁・図3―78）。マイコン側の作戦は、

① 強気でいくか、弱気でいくか。

② 直前の回に出した自分の手と同じにする

か、反対の手にするか。

③　直前の回の相手の手と同じにするか、反対の手にするか。

の6つの要素によって決まります。もちろん、実際のジレンマではもっと多くの要素が作戦の決定にかかわってくるでしょう。161頁の「囚人の自問」でみたようにです。

```
カ゛クシュウ
                Q
?0              P
?1              Q
?0              P
?1              Q
?0              Q
?0              P
?1
```

図3－76　学習が始まる

ここでは、6つの要素についてだけ考え、結果をみて、それぞれの要素が作戦の決定にかかわる重みをかえていくことにしました。

たとえば、いつも強気で成功していたなら、①の「強気」の占める比率が大きくなります。そして「強気」の作戦は、「2」ですから「2」を出力する確率が大きくなるでしょう。

```
?1              P
?1              P
?0              P
?1              P
?1              P
?1              P
?1              P
?1
```

図3－77　学習がほぼ完成

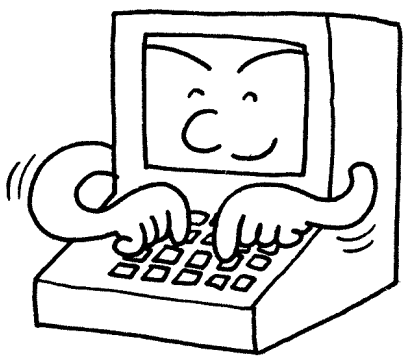
図3－78　ジレンマ・ゲームのプログラム

```
100 REM PRISONER'S
                    DILEMMA
110 DIM P(2, 2),
                    Q(2, 2)
120 FOR I=1 TO 2
130 FOR J=1 TO 2
140 READ P(I, J),
                    Q(I, J)
150 NEXT J
160 NEXT I
170 READ T
180 CLEAR
190 PRINT "ｼｭｳｼﾝ ﾉ
                    ｼﾞﾚﾝﾏ"
200 PRINT
210 PRINT "ｱﾅﾀ ﾓ
   ﾜﾀｼ ﾓ ｻｸｾﾝ 1, 2 ﾉ"
220 PRINT "ﾄﾞﾁﾗｶ
        ｦ ﾄﾙ ｺﾄ ｶﾞ
                    ﾃﾞｷ ﾏｽ"
230 PRINT
240 PRINT "ｱﾅﾀ ﾉ
 ﾄｸﾃﾝ      ﾜﾀｼ ﾉ ﾄｸﾃﾝ"
250 PRINT TAB(5);
   "1   2";TAB(17);"1   2"
260 PRINT "ﾜ";TAB
                    (12);"ｱ"
270 PRINT "ﾀ 1 ";
          P(1, 1);P(2, 1);
280 PRINT TAB(12);
          "ﾅ 1 ";Q(1, 1);
                    Q(1, 2)
290 PRINT "ｼ 2 "
          ;P(1, 2);P(2, 2);
300 PRINT TAB(12);
          "ﾀ 2 ";Q(2, 1);
                    Q(2, 2)
310 PRINT
320 PRINT "ｻｸｾﾝ 1 ﾊ
          ﾖﾜｷ ﾉ ｻｸｾﾝ"
330 PRINT "ｻｸｾﾝ 2 ﾊ
   ﾂﾖｷ ﾉ ｻｸｾﾝ ﾃﾞｽ"
340 PRINT
350 PRINT "ﾜﾀｼ タチ
   ｶﾞ ｿﾚｿﾞﾚ";T;" ﾃﾝﾆ"
360 PRINT "ﾅﾚﾊﾞ
   ｹﾞｰﾑ ﾊ ｵﾜﾘ ﾃﾞｽ"
370 REM INITIALIZE
380 DIM C(3, 2), E(3),
                    D(3, 2), F(3)
390 LET R=0
400 LET S=0
410 LET X=0
420 LET Y=0
430 FOR I=1 TO 3
440 FOR J=1 TO 2
450 LET C(I, J)=0
460 NEXT J
```

```
470 NEXT I
480 LET C(1, 1)=1
490 REM DECISION
500 LET E(1)=1
510 LET E(2)=1
520 IF Y=1 THEN
        LET E(2)=2
530 LET E(3)=1
540 IF X=2 THEN
        LET E(3)=2
550 LET L=0
560 LET Y=1
570 LET K=RND(1)
580 FOR I=1 TO 3
590 FOR J=1 TO 2
600 LET
        L=L+C(I, J)
610 IF K>L THEN 650
620 IF E(I)=J THEN
        LET Y=2
630 LET N=I
640 GOTO 670
650 NEXT J
660 NEXT I
670 REM DISPLAY
680 INPUT X
690 LET
        R=R+P(X, Y)
700 LET
        S=S+Q(X, Y)
710 PRINT Y;
720 PRINT TAB(5);
        "アナタ ";R;
730 PRINT TAB(14);
        "ワタシ ";S
740 IF R<30
        THEN 790
750 IF S<30
        THEN 790
760 PRINT
770 PRINT
    "ケ゛ーム セット!!"
780 STOP
790 REM MODIFY
800 LET A=Q(X, Y)
        *0.1
810 IF Y=2
        THEN 850
820 FOR I=1 TO 3
830 LET E(I)=3-E(I)
840 NEXT I
850 FOR I=1 TO 3
860 LET F(I)=0
870 FOR J=1 TO 2
880 LET D(I, J)=0
890 NEXT J
900 LET D(I, E(I))
        =A
910 NEXT I
920 LET F(N)=A
930 IF A>0
        THEN 1010
```

```
1030 FOR J=1 TO 2
1040 LET C(I, J)=(1-A)*
          C(I, J)+D(I, J)/5
1050 LET C(I, J)=
                C(I, J)+F(I)
1060 NEXT J
1070 NEXT I
1080 GOTO 490
1090 DATA 1, -2, 2, -1
1100 DATA 1, 2, -2, -1
1110 DATA 20
1120 END
```

```
 940 LET A=-A
 950 FOR I=1 TO 3
 960 LET F(I)=
                F(I)+A
 970 FOR J=1 TO 2
 980 LET D(I, J)=
                D(I, J)+A
 990 NEXT J
1000 NEXT I
1010 FOR I=1 TO 3
1020 LET F(I)=F(I)/5
```



BASIC FOR MICROCOMPUTERS

第 4 章

実用計算のプログラムと使い方

売上げ集計

私たちが仕事や日常生活の中で利用しようとする資料は、まったく整理されていないのが普通です。新聞の切抜帳を作っても、商品の売上げ伝票を綴じておいても、それは「まとめておいてある」というだけで、本当の整理ではありません。そのような資料から何かを得るためには、資料の必要な部分だけを抜き出し、分類して集計するという操作が必要です。

たとえば、2月4日には商品Aが3ケース売れ、2月8日には、商品Bが2ケース売れた、という伝票を何百枚集めても、これからの1ヵ月に商品AとBがどのくらい売れるかという予測が、すぐに立てられるわけではありません。伝票を繰っていくうちに「だいたい」の傾向はわかるかもしれませんし、また「だいたい」の仕入れ量を決めても、うまくいくときは、うまくいくかもしれません。

しかし、それでは資料を十分に活用しているとはいえません。

この資料の整理に、マイコンを利用することを考えてみましょう。マイコンが得意とするのは、「分類」と「集計」の処理です。ここでは、特に、「集計」を取り上げます。

客観的な資料を得ることさえできれば、マイコンによる集計は、有益な結果をもたらしてくれるはずです。

これから説明するプログラムは、主に、会社や商店での売上げ集計と、集計結果のグラフ表示を目的としています。説明の便宜のために、例を自動車会社での売上げ集計にとりました。プログラムがそのまま、衣料品、食料品などの

売上げ集計に使えることはもちろんです。ここがマイコンの長所でもあり、短所でもあるのですが、プログラムが「このデータは何のデータであるか」という判断を行うことはありません。同じ仕事であれば、データの性質いかんにかかわらず働いてくれます。

集計

ある自動車会社は、いくつかの販売店に販売を任せています。販売店では、自動車が1台売れるたびに、売れた車の種別などを記入した売上げカードを本社に送ります。本社の販売部の机の上には、この売上げカードがうず高く積まれるわけです。

この売上げカードの1ヵ月分をもとにして、1ヵ月間に、どの種類の車がよく売れたかを集計することにします。

1　普通乗用車
2　ライトバン
3　小型トラック
4　大型トラック
5　バス

の5種類を、この会社で作っているとします。ある月間の売行きのパターンがわかれば、次の月間の生産計画に役立ちますから、「集計」は重要な仕事です。

集計計算

マイコンを使ってこの集計を行うには、どのような方法で売上げカードの記事を入力し、集計の結果をどういう形式で出力するかを決めなければなりません。

入力の方法

売上げカードの記事は、1枚ずつめくりながら、「1」から「5」までの番号で入力すること

にします。つまり、マイコンから「?」が出てくるたびに売上げカードをみて、「ライトバン」と書いてあれば「2」のように入力するのです。この作業を売上げカードの枚数分だけ続けます。

集計表の形式

図4－1には、車の種別ごとに、売れた台数と全体に占める割合(パーセンテージ)を出力するようにしてあります。もちろん、全部で何台売れたかも、「ケイ」で出力します。この合計は、パーセンテージを計算するときにも必要です。

プログラムの手順

このような作業を行うプログラムは、

① 入力に応じて、各車種の売上げ台数を数える。

② 全体で何台売れたかを求める。

③ 各車種について、売れた台数と、全体に占める割合を出力する。

という3つの仕事をしなければなりません。

②の仕事は、①の仕事の結果を必要とします。③の仕事は、①と②の仕事の結果を使います。仕事の結果を残しておくためにプログラムでは変数を使います。

シュウケイ	ケッカ	
1	27	46.5517
2	8	13.7931
3	12	20.6896
4	8	13.7931
5	3	5.17241
ケイ	58	100.0 %

図4－1　集計結果の出力

①の仕事の結果は、配列変数Aを、

DIM A(5)

と宣言して、これに入れておきます。仕事①のプログラムが終ったとき、A(1)には、種別1の車（普通乗用車）の売れた台数、A(2)には、種別2の車の売れた台数が入っている、という具合です。仕事②の結果は、変数Sに入れておきます。Sは、全体で売れた台数ですから、

S＝A(1)＋A(2)＋A(3)＋A(4)＋A(5)

という式で計算されます。そして、仕事③では、種別K（Kは1から5までのどれか）の車の売り上げ台数の全体に占める割合が、

A(K)÷S×100　％

のように計算されるわけです。

これらの変数を用いて、仕事①～③を書直してみましょう。

①　データKが入力された回数を数えて、A(K)に入れておく。

②　SはA(1)、A(2)、……、A(5)の和。

③　A(K)が、Sの何パーセントか、を出力する。

それぞれの手順は、もう少し詳しく記述しなければなりません。①は、

①′—①A(1)、A(2)、……、A(5)をすべて0にする。

①′—②入力データがKならA(K)を1だけ増加させる。これを入力データの数だけ続ける。

のようになります。

入力データの終りを示すために、「0」を入力してやります。プログラムでは、入力データとして「0」が入ったところで、次の仕事に移るようにします。①′の手順を178頁の図4—2に示しました。

このフローチャートにしたがってプログラム

を作ると図4—3のようになります。120～140FOR—NEXTループで、A(1)、……、A(5)をすべて0にし、150で、データKを入力します。このとき画面には「?」が出て、入力を要求しているはずです。Kが0なら、160から、190に抜け出しますが、そうでなければ、180から150に戻って、次のデータ入力を待ちます。これで、手順①'が完成しました。

手順②'は、手順①'で定義されたA(1)、A(2)、……、A(5)の総和Sを求める手順です。フローチャートを書くまでもないでしょう。図4—4がそのプログラムです。5回の繰返しに、FOR—NEXTループを用いるのは大げさかもしれませんが、変更、修正が容易になりますから、

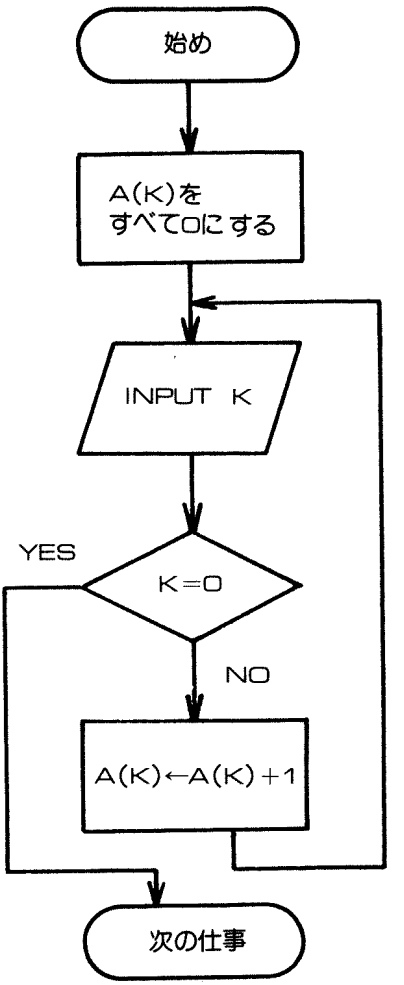


図4－2　集計の手順(フローチャート)

なるべくループを使うべきなのです。

結果の出力

手順③のプログラムを書くためには、もう1度図4—1を見返す必要があります。タイトル「シュウケイ　ケッカ」の出力を含めて、180頁の図4—5を書いてみましょう。

プログラムは、181頁の図4—6です。

これで、やっと各手順ができあがりました。全体の手順の進行は、180頁の図4—7、プログラムは図4—8にまとめます。

```
100 REM シュウケイ
110 DIM A(5)
120 FOR I=1 TO 5
130 LET A(I)=0
140 NEXT I
150 INPUT K
160 IF K=0 THEN 190
170 LET A(K)=A(K)+1
180 GOTO 150
190 REM ツキ゛ ノ シコ゛ト
```

図4－3　集計のプログラム

```
200 REM ソウ ウリアケ゛ タ゛イスウ
210 LET S=0
220 FOR I=1 TO 5
230 LET S=S+A(I)
240 NEXT I
```

図4－4　総和を求めるプログラム

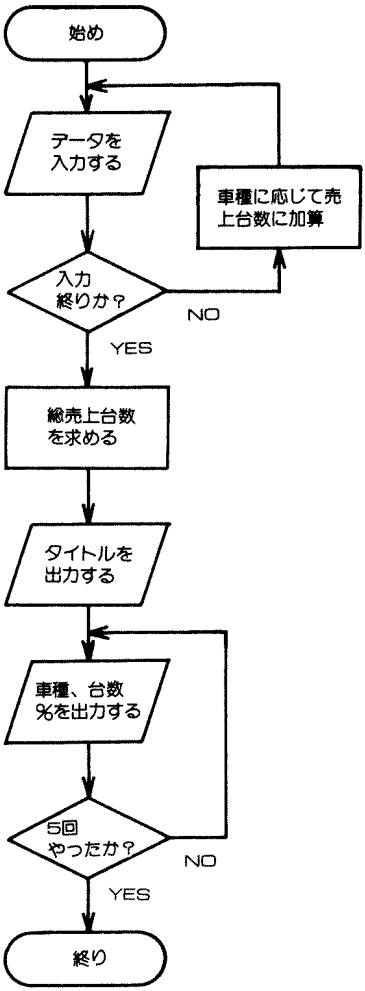


図４－７　全体のフローチャート

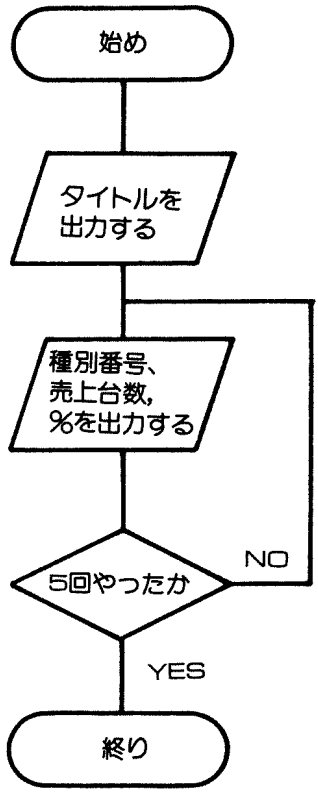


図４－５　結果の出力のフローチャート

```
300 REM ケッカ ノ シュツリョク
310 PRINT "シュウケイ ケッカ"
320 PRINT
330 FOR I=1 TO 5
340 PRINT I;TAB(5);A(I);TAB(10);A(I)/S*100
350 NEXT I
360 PRINT
370 PRINT " ケイ";TAB(5);S;TAB(10);" 100.0 %"
```

図4－6　結果の出力のプログラム

```
 10 REM シュウケイ
 20 DIM A(5)
 30 FOR I=1 TO 5
 40 LET A(I)=0
 50 NEXT I
 60 INPUT K
 70 IF K=0 THEN 100
 80 LET A(K)=A(K)+1
 90 GOTO 60
100 LET S=0
110 FOR I=1 TO 5
120 LET S=S+A(I)
130 NEXT I
140 PRINT "シュウケイ ケッカ"
150 PRINT
160 FOR I=1 TO 5
170 PRINT I;TAB(5);A(I);TAB(10);A(I)/S*100
180 NEXT I
190 PRINT
200 PRINT " ケイ";TAB(5);S;TAB(10);" 100.0 %"
210 STOP
220 END
```

図4－8　全体のプログラム

ヒストグラム

図4—8（181頁）のプログラムだけでも十分役に立ちますが、各車種の売上げ台数をグラフに表示すると、どの車種が一番売れたかなどがひと目でわかって便利です。グラフにも折線グラフ、棒グラフ、帯グラフなどいろいろあります。このうち、マイコンのCRTに表示しやすいのは棒グラフ（ヒストグラム）です。CRTの画面を方眼紙（たとえば16×32）にみたててグラフを書けるからです。

ヒストグラムの一番簡単な書き方は、車種3番の売上げ台数12に対して「＊」や、ほかの文字・記号を同じ数12だけ画面に並べる方法です。この方法では、画面のタテ・ヨコに入りきる数（16と32）以上の数は表示できません。

その点を少し工夫してみなければなりません。

```
ヒスト　ク゛ラム

 １    ＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
 ２    ＊＊＊＊＊
 ３    ＊＊＊＊＊＊＊＊
 ４    ＊＊＊＊＊
 ５    ＊＊
```

図4－9　図4－1の集計のヒストグラム

私たちがグラフから読みとりたいのは、それぞれの車種の細かい売上げ台数ではなく、全体でどの車種が一番多く売れたかなどのパターンです。車種1の方が車種2より多く売れたのに、グラフでは車種2の方が多いようにみえるなど

ということがなければ、グラフとしてはそれで十分でしょう。正しい台数を知りたければ集計表の方をみればよいわけです。

ヒストグラムの形式

ここでは、図4—9のような表示を出すヒストグラムのプログラムを作ってみましょう。図4—9のグラフは、図4—1の集計をそのままグラフにしたものです。

車種1は、図4—1によると27台ですが、図4—9では20個の「＊」で表示されています。ほかの車種でも図4—1の台数と図4—9の「＊」の数は違っています。ただ割合だけは同じです。つまり車種3は、車種1の半分弱だという点は、図4—1の正しい売り上げ台数からも、図4—9のヒストグラムからも読みとることができます。このプログラ

```
シュウケイ   ケッカ

1          13        26.5306
2          6         12.2448
3          8         16.3265
4          15        30.6122
5          7         14.2857

ケイ       49        100.0 %
```

図4－10　集計の例

```
ヒスト  ク゛ラム

1   ＊＊＊＊＊＊＊＊＊＊＊＊＊
2   ＊＊＊＊＊＊
3   ＊＊＊＊＊＊＊＊＊＊
4   ＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
5   ＊＊＊＊＊＊＊＊＊
```

図4－11　図4－10の集計のヒストグラム

ムのミソは、図4—10の集計が図4—11のようになることです。

図4—10の集計では最大数が15で、図4—10の最大数27の半分より少し多いくらいですから、ヒストグラムを書くと、ほとんどが「＊」10個以下になると思うかもしれません。ところが、画面10では、最大数の15に対応するところでは「＊」が20個になっていますし、ほかでも割合に応じて「＊」がもっと多く出ています。

つまり、

① データの中で最大の数に対しては、いつも「＊」を20個出力する。

② ほかのデータでは、最大データに対する割合を計算して「＊」を出す。

ということになっているのです。

このようにするのは、グラフがどんなデータに対してもきれいに出るようにしたいからです。ある月は、車の売行きがよくて「＊」が画面をとび出してしまい、ある月は、あまり売れなくて、「＊」が1つも出ないというのでは不便です。グラフの最大値をいつも「＊」20個になるようにしておけば、そのような心配はいらないわけです。ただ、各月ごとにグラフを書くと、「＊」1個に相当する台数が月によって違うことになりますから、注意しなければなりません。

プログラムの作成

それでは、プログラムの作成にかかりましょう。まず、仕事は、

① データの最大値を求める。

② データの最大値に対しては、「＊」20個だから、「＊」1個が売上げ何台分かを計算する。

③ 各データが「＊」何個分かを計算して、

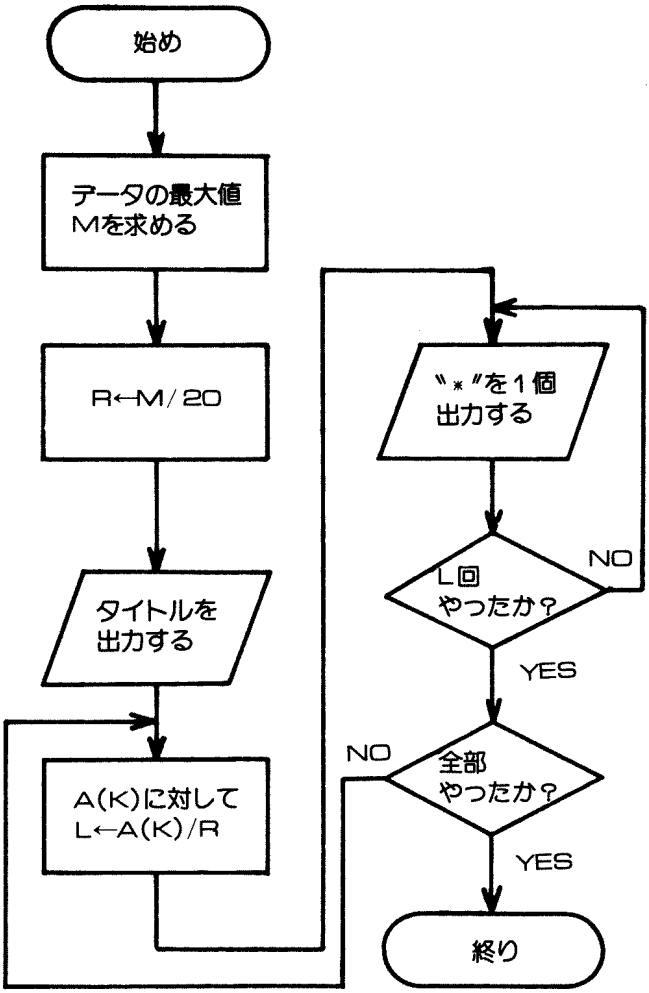


図4－12　ヒストグラムのフローチャート

画面に出力する。

の3つです。

データは、A(1)、A(2)、……、A(5)という配列変数に入れてあるとします。その中で1番大きい数（最大値）を変数Mに入れます。Mは、「＊」20個分の売上げ台数です。Mを20で割れば、「＊」1個分の台数がわかります。それを変数Rに入れましょう。

R＝M÷20

ある車種の売上げ台数A(K)を、このRで割れば、A(K)が「＊」何個分かを計算することができます。それを、Lとすれば、

L＝A(K)÷R

で、L個の「＊」を出力することになります。

Lは、整数とは限りませんが、

L＝4.82456

のような時は、「＊」を4個出力します。

```
200 REM ヒスト ク゛ラム
210 LET R=M/20
220 PRINT "ヒスト ク゛ラム"
230 PRINT
240 FOR I=1 TO 5
250 PRINT I,
260 LET L=A(I)/R
270 FOR J=1 TO L
280 PRINT TAB(J+3);"＊";
290 NEXT J
300 PRINT
310 NEXT I
```

図4－13 ヒストグラムのプログラム

変数の名前が決ったところで、手順をまとめてみましょう。

①′ A(1)、A(2)、……、A(5)の中の最大値を求め、変数Mに代入する。

②′ 「＊」1個分の台数RをR＝M÷20で求める。

③′ A(K)に対して、「＊」の数LをL＝A(K)÷Rで求め、その数だけ「＊」を出力する。これをA(1)、A(2)、……、A(5)について行う。

フローチャートも書いてみます（185頁・図4—12）。

手順②′、③′のプログラムは簡単です。図4—13に示しました。タイトルの「ヒスト　グラム」も出力するようになっています。

図4—13で注意してほしいのは、250、280、300のPRINT文です。ヒストグラムの1行の出

```
                                    ]  実行前

 3  _                               ]  250実行後

 3     * *_                         ]  280実行

 3     * * * * * * * * * *_         ]  280実行後

 3     * * * * * * * * * *          ]
                                    ]  300実行後
_
↑
カーソル
```

図4—14　グラフを書くときのカーソルの動き

力は、この3つのPRINT文が手分けして行います。

まず、250のPRINT文が、車種番号を出力して、カーソルを4桁目にすすめます（図4－14）。PRINT文の最後は「;」ですから、カーソルが次の行に移ることはありません。280のPRINT文は、L回実行されますが、そのたびに、「*」を1つ出力して、カーソルを1つすすめます。

「;」で終っていて、やはりカーソルを次の行に送ることはしません。

カーソルを次の行の先頭に持ってくるのは、

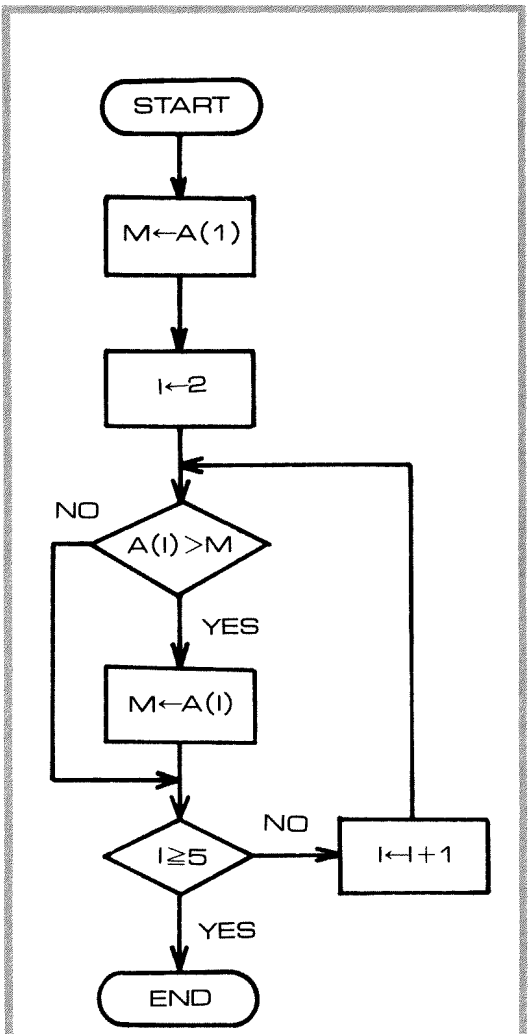


図4－15　最大値を求めるフローチャート

```
100 REM サイタ・イチ
110 LET  M=A(1)
120 FOR I=2 TO 5
130 IF A(I)>M THEN LET M=A(I)
140 NEXT I
```

図4－16　最大値を求めるプログラム

「＊」をL個出力したあとの300のPRINT文の仕事です。280番でTABを使っていますが、これは、使わずに済むかもしれません。ただ、TABの引数に変数を入れる手法は、心得ておいてください。

TAB(X)

は、XがこのTABを使う時に3であればTAB(3)と同じ意味ですし、Xが10であればTAB(10)と同じ意味になります。

最大値を求める

手順①は、いくらか厄介です。「最大値を求めろ」という命令がないからです。そのために、次のような手順を考えます。

①′—①　A(1)が最大値かもしれない。まず、MにA(1)を代入する。

①′—②　MとA(2)を比べてみる。A(2)の方が大きければA(2)が最大値かもしれないから、MをA(2)にする。A(2)がMより大きくなければA(2)は最大値ではないからMはそのまま。

①′—③　MとA(3)を比べて同じことをする。これをA(4)とA(5)についても行う。全部比べ終ったら、MはA(1)、……、A(5)の最大値になっているはず。

この手順をディティール（詳細）フローチャートにしてみます（図4—15）。

フローチャートができれば、プログラムはすぐに書けます（図4—16）。図4—13と図4—16

図4－17　集計とグラフを出力するプログラム

```
 10 REM シュウケイ ト ｸﾞﾗﾌ
 20 DIM A(20)
 25 CLEAR
 30 PRINT "ﾌﾟﾝﾙｲ ﾊ
          イクツ ﾃﾞｽｶ"
 40 INPUT C
 50 FOR I=1 TO C
 60 LET A(I)=0
 70 NEXT I
 80 PRINT "ﾃﾞｰﾀ ヲ
          ﾄﾞｳｿﾞ"
 90 INPUT K
100 IF K=0 THEN 170
110 IF K<0 THEN 150
120 IF K>C THEN 150
130 LET A(K)=A(K)+1
140 GOTO 90
150 PRINT "ﾏﾁｶﾞｲ
          ﾃﾞｽ"
160 GOTO 90
170 LET S=0
180 FOR I=1 TO C
190 LET S=S+A(I)
200 NEXT I
210 CLEAR
220 PRINT "シュウケイ
          ケッカ"
230 PRINT
240 FOR I=1 TO C
250 PRINT I;TAB(5);
          A(I);
260 PRINT TAB(10);
          A(I)/S*100
270 NEXT I
280 PRINT
290 PRINT " ケイ";
          TAB(5);S;
300 PRINT TAB(10);
          " 100.0 %"
310 LET M=A(1)
320 FOR I=2 TO C
330 IF A(I)>M THEN
          LET M=A(I)
340 NEXT I
350 CLEAR
360 LET R=M/20
370 PRINT "ヒスト
          ｸﾞﾗﾑ"
380 PRINT
390 FOR I=1 TO C
400 PRINT I;
410 LET L=A(I)/R
420 FOR J=1 TO L
430 PRINT TAB(J+3)
          ;"*";
440 NEXT J
450 PRINT
460 NEXT I
470 STOP
480 END
```

をあわせて、ヒストグラムを書くプログラムはできあがりです。

集計とグラフのまとめ

ここで集計とグラフのプログラムをまとめてみます。図4—8、13、16をつなげばよいのですが、少し一般性を持たせてみます。自動車の種類は5種類とは限りませんから、種類の数も入力します。また、決まった種類以外の数を入力したとき（5種類と決めたのに「6」を入力したとき）は「マチガイ」と出力して、もう一度入力できるようにします。これで、かなり実践的になるはずです。全体のプログラムを図4—17に示しました。

プログラムの使い方

「RUN」を入力すると、「ブンルイ ハ イクツ デスカ」と出力してきますから、3種類なら「3」と入力します。「データ ヲ ドウゾ」が表示されたあと、「?」が出てくるたびに、データを入力します。データが終って、「0」を入力すると、計算を始め集計結果とヒストグラムを出力してくれるはずです。

ファイルの使い方

次のプログラムを紹介する前に、第2章で言葉だけ出てきた「ファイル」について、説明することにします。このファイルを使いこなすことができなければ、マイコンの持っている能力を十分に活用することはできません。

データとアイテム

「172」、「65」、「168」、「54」という4つの数字があったとします。これらの4つの数字は、ただの数で、データでも何でもありません。これらの数が、「データ」と呼ばれるためには、たとえ

ば、「172cmは、A君の身長」というように、「ある人(物)についての何か」という情報が必要です。「身長」とか、「体重」とかいう名前は、数値に「意味」を与えます。数値(あるいは、「セタガヤ」のような文字の列)に意味が与えられて、「体重54kg」のように結びつけられたものをアイテムといいます。「体重」をアイテムと呼んで、「54」を、そのアイテムの「値」ということもあります。

アイテムとエントリ

A君に関するデータには、「身長」とか「体重」とか、さまざまな種類があります。つまり、A君を記述するために、いくつかのアイテムが使われるのです。「A君に関する記述」という点では共通した、いくつかのアイテムの集まりをエントリといいます。「A君に関する記述」が、エントリそのものだともいえます。

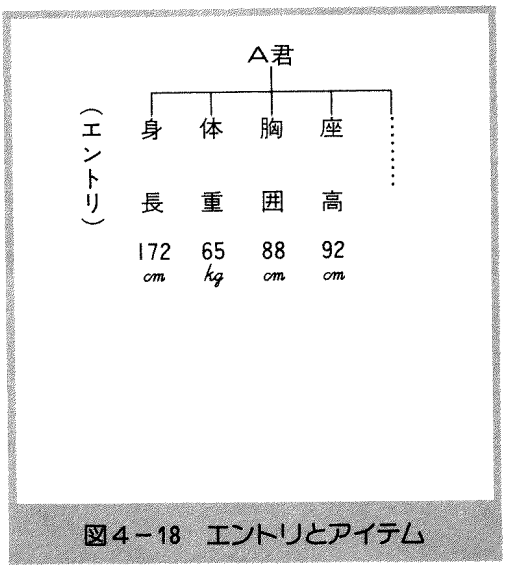


図4－18　エントリとアイテム

この関係を図示すると、図4―18のようになります。

エントリとファイル

A君についてだけでなく、B君やC君についてもやはり、身長や体重などのアイテムを集めてエントリを作ります。A君を記述するエントリ、B君を記述するエントリなど、いくつかの

エントリが集まります。このように、いくつかのエントリを集めたものがファイルです。

ファイルは、たとえば、「ある学校で、生徒の身体測定を行なった結果」という意味を持つことになるでしょう。

まとめるとファイルは、いくつかのエントリで構成され、エントリは、いくつかのアイテムで構成されるというわけです。この関係を図4—19に示します。

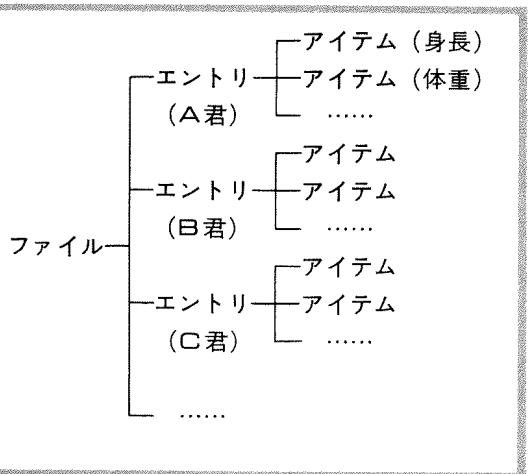


図4—19　ファイル・エントリ・アイテムの関係

カセット・テープとファイル

今までに説明したファイルは、結局、データの集まりに、勝手な意味をつけただけのものですから、それらのデータが、どこに、どのような形式で記録されているかは、問題になりません。

ノートにデータを書込んで、アイテム・エントリの関係を図で表わしたものもファイルですし、マイコンの記憶部の中に電気的な形式で保存されたものもファイルです。

しかし、普通、マイコンで扱うファイルといえば、カセット・テープやフロッピーディスクに記録されたデータの集まりを意味します。ここでは、特にカセットテープ上のファイルを考えることにしましょう。

カセット・テープ・ファイルの作成

カセット・テープにこの「ファイル」を作るためには、カセット・テープにデータを「書き入れ」たり、データを「読み出し」たりする機能が、マイコンに用意されていなければなりません。

カセット・テープを使ってデータを読み書きする機能は、どのマイコンにもあるというわけではありません。それに、読み書きする方法は、マイコンによって違っています。

本書では、ファイルの扱い方に深入りしませんから、

```
PRINT FILE ～
INPUT FILE ～
```

という2つのベーシックの文だけを使うことにします。この2つの文が、そのまま使えるマイコンは、残念ながらありません。マイコンの機種に応じて、キーワード（PRINT FILE、INPUT FILE）を置きかえる必要があります。

さらに、ファイル制御のための文（たとえば、OPEN、CLOSE）が用意されているマイコンでは、それを使うことも必要です。

本書の例では、テープを先頭に戻してから、

```
PRINT FILE "ﾌｧｲﾙ"
```

という文を実行すると、「ファイル」という文字が、テープに書込まれ、もう一度テープを巻戻して

```
INPUT FILE S$
```

を実行するとS$という文字型変数に、「ファイル」という文字が読込まれるものとしました。

数値型の場合も、

```
PRINT FILE K（Kは48）
```

を実行すると、「48」がテープに書込まれ、

```
INPUT FILE N
```

を実行すると、変数Nに「48」が代入されるわけです。

これらの文を実行する間には、テープ巻戻しの操作を手動で行なわなければならないというような面倒さもありますが、それは仕方のないことです。

在庫問合せ処理

在庫の有無の問合せには、「商品Aが在庫しているか」など商品名を指定して、その在庫を問合わせてくる場合のほかに、「条件Bをみたす商品があるか」など商品名を指定せずに、在庫を問合わせてくる場合があると思います。商品名を指定してきた場合は、問合せを受けた時点の在庫商品リストの「商品名」欄だけに目を通せばよいのですから、答えるのも容易でしょう。取扱う商品の種類が固定されていて、在庫管理がうまく行われ、常に適正在庫が期待できるような会社の場合は、在庫リストなどみなくても、「商品Aは、たしかに在庫しています。ただ、発注量によりましては、納入期日に余裕をいただかなければなりません。」と、その場で答えることができるはずです。

ところが、「ある条件をみたす在庫」という問合せは、いくらか厄介です。商品名を指定しない在庫問合せは、小売店なら、むしろ普通です。「こんな形で、これくらいのお値段のものがないかしら」という質問に、即座に答えられるか否かは、小売店のサービスの重要な部分とさえいえるでしょう。一般の小売店では、商品に関する質問に即座に答えるという課題を、担当者の「熟練」によって解決しています。しかし、その熟練は、短期間で得られるものではありません。

不動産や、「情報」を商品として扱う場合は、困難がさらに大きくなります。家屋、土地、デ

ータ、ニュースは、同じものが大量にあるのではなく、1つ1つ異っているからです。このような「多品種少量(単一)在庫」を効率的にみつけ出し、問合せに迅速に答えるのは容易ではありません。しかも、これらの在庫は、次々に内容が変化していきます。家は売れてしまえば、在庫リストから除かなければなりませんし、ニュースは、次の日には無価値になってしまいます。ある程度以上の大きさの在庫は、管理が非常にむづかしくなるでしょう。

そこでマイコンに登場してもらおうというわけです。

在庫ファイル

エントリは、原則として在庫の属性を示すアイテムだけからなっています。最初のアイテムは、在庫番号とし、最後のアイテムは、ファイル外への参照番号とします。ファイルに入れにくいデータ(たとえば、家屋の見取図)は、この参照番号によって検索します。

1 在庫番号
2 属性1
NN−1：属性N−2参照番号

アイテム2〜N−1は、在庫の属性を書込んだものです。

2 アパート名
3 住所
4 利便
5 間取
6 賃料
7 程度

のような属性をあげておけば十分でしょう。こ

れから、不動産業でのアパート紹介を例にとって説明を進めることにします。

ファイル設計

エントリは、9つのアイテムからなります。アイテム2、3には、文字型のデータを使い、そのほかのアイテムは、数値型にしましょう。アイテム4、5、7は、内容を適当にコード化します。

1　在庫番号
2　アパート名
3　住所
4　利便
　1　駅から10分未満
　2　駅から10分以上30分未満
　3　駅から30分以上
5　間取
　1　1DK
　2　2DK
　3　3DK
6　賃料（月額）
7　程度
　1　上
　2　中
　3　下
8　参照番号

というのが、エントリの詳細です。参照番号は、間取図や付近の地図などの資料を保管してある棚の番号にでもすればよいわけです。

ファイルの作成

まず、手持の物件を、すべてファイルに書込まなければなりません。198頁の図4—20が、ファイル作成のプログラムです。

```
ﾆｭｳｼｭﾂ ﾃﾞｰﾀｼｭｳ
```

と出てきたあとアイテム1、2、3、4、5、6、7、8の順に入力していきます（199頁・図

```
100 REM FILE CREATE
110 PRINT "ニュウリョク ヲ ト゛ウソ゛"
120 INPUT N
130 PRINT FILE N
140 IF N=0 THEN 240
150 FOR I=1 TO 2
160 INPUT K$
170 PRINT FILE K$
180 NEXT I
190 FOR I=1 TO 5
200 INPUT N
210 PRINT FILE N
220 NEXT I
230 GOTO 110
240 STOP
250 END
```

図4—20　ファイル作成のプログラム

4—21)。

すべての物件を入力し終ったら、

ニュウリョク　ト゛ウソ゛

のあとに「0」を入力します。

そのあと、テープを巻取って、保管しておきます。

在庫捜し

不動産屋に、お客が入ってきて、「アパートを捜しているんですが」といいます。不動産屋の主人は「どのあたりで」と聞返します。「この近くにあればいいんだけれどね」、「今頃はアパートの空室も少ないんですよ。どのくらいの広さをお考えですか」、「2DKかな」、「御予算は」、「5万くらいだね」、「ちょっと待ってくださいよ。駅から少し遠くてもかまいませんか」、「近いにこしたことはないけれど、まあ、10分や20

```
ニュウリョク　ヲ　ト゛ウソ゛
？１
？ケイナンコーホ゜
？セタカ゛ヤ
？１
？２
？５２０００
？１
？１８
ニュウリョク　ヲ　ト゛ウソ゛
？２
```

図４－21　ファイル作成のときの画面

分歩くくらいなら」、「ここなんかどうでしょう。2DKで5万2千円、少し遠くなりますけどね。静かなところで、学校も近いはずですよ」、「いや、学校が近くても仕方がないけど、5万2千円か、部屋を見せてくれるかな」、「どうぞ。今、案内しましょう」というように、話が進みます。

この手順を、全部マイコンに任せてしまおうというのです。客は、不動産屋の主人と応対する代りに、マイコンに向って自分の希望を伝えます。マイコンは、在庫ファイルの中から、客の希望にあうものを探し出して、画面に表示します。客は、気に入ったものがあれば、不動産屋の主人に、「ここがいい。みてきたいから、道順を教えてくれ」頼むわけです。何もかもセルフ・サービスになっては面白くないというかもしれませんが、能率だけはよいはずです。

このような在庫捜しの手法は、アパートに限

らず、一般の商品選びにも応用することができます。

プログラムを図4—22に示します。次に、その使い方を説明することにしましょう。

プログラムの使い方

カセット・テープ・ファイルは、シーケンシャル・ファイルですから、検索の手法も決まっています。エントリ（レコード）ごとに読み出すのです。

条件は、前もって入力しておきます。そして読み出したエントリのアイテム（フィールド）の値と比較して、そのエントリを捨てるかどうかを決めます。条件に合ったエントリは覚えておいて、いくつかたまったら画面に表示するわけです。

アパート捜しの例

```
120 INPUT FILE Y
130 INPUT FILE F
140 INPUT FILE G
150 IF P<>X THEN 70
160 IF Q<Y THEN 70
170 PRINT A$
180 PRINT B$
190 PRINT "リヘ゛ン:";E
200 PRINT "ヤチン  :";Y
210 PRINT "テイト゛ :";F
220 PRINT G
230 PRINT
240 GOTO 70
250 STOP
```

```
 10 REM FILE
        SCANNING
 20 PRINT "マト゛リ
    ノ コ゛キホ゛ウ ハ゛"
 30 INPUT P
 40 PRINT"オヤチン ノ
    コ゛キホ゛ウ ハ゛"
 50 INPUT Q
 60 PRINT
 70 INPUT FILE N
 80 IF N=0 THEN 250
 90 INPUT FILE A$, B$
100 INPUT FILE E
110 INPUT FILE X
```

図4－22　ファイルを調べるプログラム

まず、マイコンから、いくつかの質問をしてきますから、数値を入力して、質問に答えてください。

2DKが希望なら、

ﾏﾄﾞﾘ ﾉ ｺﾞｷﾎﾞｳ ﾊ

と出たあとに、「2」を入力します。全部の質問に答え終ると、ファイルの走査が始まり、希望の条件に合ったアパートの名と住所を表示してくれます。希望が、虫のよすぎるものだったりすると、

ｷﾞﾝﾈﾝ ﾃｽｶﾞ ｺﾞｷﾎﾞｳ
ﾉ ｼﾞｮｳｹﾝ
ﾃﾞﾝ ﾜﾜｶﾆ ﾏｾﾝ

というメッセージがでてくるようにしてもよいでしょう。

図4—23にプログラムの実行例を示します。

```
ﾏﾄﾞﾘ ﾉ ｺﾞｷﾎﾞｳ ﾊ
?2
ｵﾔﾁﾝ ﾉ ｺﾞｷﾎﾞｳ ﾊ
?48000

ｱｵｷｿｳ
ｻｸﾗｼﾝﾏﾁ
ﾘﾍﾞﾝ:  1
ﾔﾁﾝ  :  47000
ﾃｲﾄﾞ :  2
  23
```

図4－23　プログラムの実行例

カレンダー

マイコンを入手したら、やはり一度は作ってみたいプログラム、そして一度作ると、あとで自由に使うことができる、そんなものとして、まず考えられるのがカレンダーを作るプログラムでしょう。

曜日を調べるプログラム

最初からカレンダーを作ろうとは考えないで、まず、ある日付を入力したときに、その日が何曜日であったかを調べるプログラムを作ってみることにしましょう。このプログラムを使えば、自分の生まれた日が何曜日であるのか、来年の元旦が何曜日か、などを簡単に調べることができます。

曜日を求める計算式

このプログラムの一番の問題点は、曜日を算出するための計算式を求めることです。曜日はつねに7日に1回同じものが繰返えされるということを利用して、計算式を出してもよいのですが、ここではよく使われる計算式を利用することにします。

この式は、図4—24のフローチャートに示されているように計算します。1月、2月と3月以降では、計算の仕方が少し違います。図の③、④の部分で明らかなように1月、2月では、その月の数に10を加え、年の数を1つ減らします。また、3月以降では逆に月の数から2を引くのです。これは全体として月を2ヵ月分前にずらしていることになります。ということは、1979年の1月は、前年の1978年11月として、また79年の5月は3月として計算します。

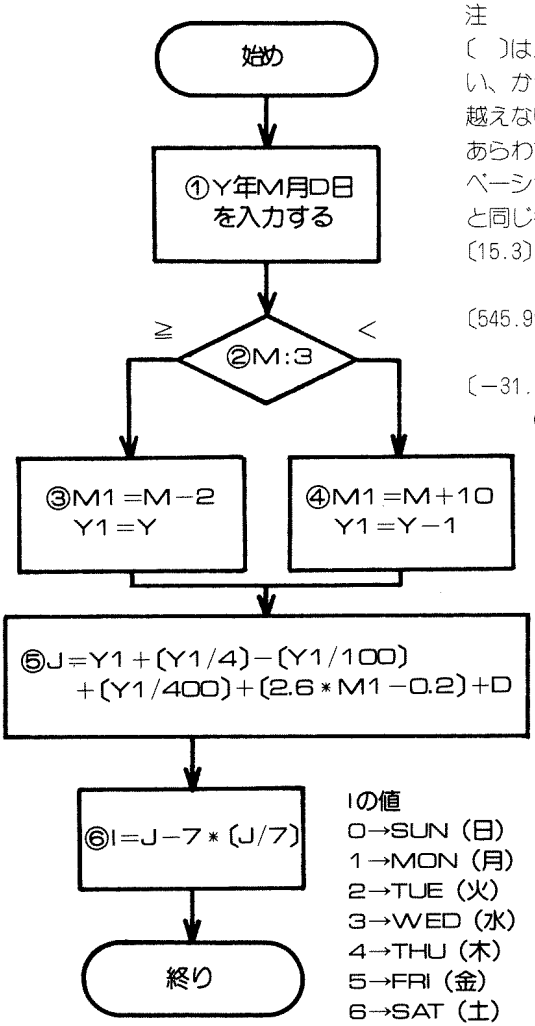


図4−24　曜日を調べるプログラムのフローチャート

1980.5.4.は何曜日か
① Y←1980, M←5, D←4　を入力
② M>3より　③に進む
③ M1=3, Y1=1980
⑤ J=1980+〔1980/4〕−〔1980/100〕+〔1980/400〕
　　+〔2.6*3−0.2〕+4
　=1980+〔495.0〕−〔19.8〕+〔4.95〕
　　+〔7.6〕+4
　=1980+495−19+4+7+4=2471
⑥ I=2471−7*〔2471/7〕=2471−7*〔353.0〕=0
　　I=0より　SUN（日曜日）

1936.2.26.は何曜日か
① Y←1936, M←2, D←26　を入力
② M<3より　④に進む
④ M1=12, Y1=1935
⑤ J=1935+〔1935/4〕−〔1935/100〕+〔1935/400〕
　　+〔2.6*12−0.2〕+26
　=1935+483−19+4+31+26=2460
⑥ I=2460−7*〔2460/7〕=2460−7*351=3
　　I=3より　WED（水曜日）

図4−25　曜日を調べる手順

なぜこのようにすると、うまく計算できるのかという理由は、少し面倒なのでここでは述べませんが、3月から1年が始まると考えると、3月〜7月までと8月〜12月までの5ヵ月づつが大の月と小の月が交互に交代しているようになること、また2月が年の終りになるために、うるう年の計算がやりやすくなることなどに関係しています。

具体的には、図4−25のようにしてJの値を求め、それを7で割ったあまり（Iの値）によって曜日を定めます。あまり（I）が0なら日曜で、6なら土曜日です。このように計算すると、図4−25の例のように、1980年5月4日はIが0になるので日曜であるとわかります。同じようにして、1936年2月26日

```
 10 REM
 20 DIM B$(6)
 30 PRINT "YEAR ?"
 40 INPUT Y
 50 PRINT "MONTH ?"          ①
 60 INPUT M
 70 PRINT "DAY ?"
 80 INPUT D
 90 REM
100 GOSUB 1000
110 REM
120 FOR K=0 TO 6
130 READ B$(K)
140 NEXT K
150 DATA SUN, MON, TUE, WED, THU, FRI, SAT
160 PRINT Y; "NEN" ;M; "GATSU" ;D; "NICHI"
170 PRINT TAB(10); "* *  " ;B$(I); "  * *"
180 STOP

1000 REM
1010 IF M>=3 THEN 1050 ②
1020 LET M1=M+10            ④
1030 LET Y1=Y-1
1040 GOTO 1070
1050 LET M1=M-2             ③
1060 LET Y1=Y
1070 LET J=Y1+INT(Y1/4)-INT(Y1/100)                    ⑤
1080 LET J=J+INT(Y1/400)+INT(2.6*M1-0.2)+D
1090 LET I=J-7*INT(J/7)⑥
1100 RETURN
```

図4－26(a)　曜日を調べるプログラム

（2・26事件の起こった日）は水曜日であったということもわかるのです。

計算するためのプログラム

さて、このような計算をするためのプログラムはどのように作ったらよいでしょうか。

図4－24のフローチャートはそのままこのプログラムに使うことができます。それにもとづいて作成したプログラムが図4－26(a)に示されています。ここでプログラムの右側に書かれた丸でかこまれた数字がフローチャートの数字に対応し

```
1979 NEN 3 GATSU 19 NICHI
          * *MON* *
1956 NEN 8 GATSU 15 NICHI
          * *WED* *
1954 NEN 9 GATSU 20 NICHI
          * *MON* *
1 NEN 1 GATSU 1 NICHI
          * *MON* *
2001 NEN 1 GATSU 1 NICHI
          * *MON* *
1979 NEN 6 GATSU 15 NICHI
          * *FRI* *
1980 NEN 1 GATSU 1 NICHI
          * *TUE* *
```

図4－26(b)　曜日を調べるプログラム

ています。

このプログラムの1000行以降は、100行のGOSUB文によってよばれるサブルーチンです。したがって、このプログラムの全体としての流れは30～80行で、年・月・日を入力したあと、このサブルーチンでこのプログラムの中心部分である計算をし（ここでIに0～6の値が入れられる）、1100行のRETURN文によって110行にもどり、結果を図4－26(b)のように出力します。

ここで、1000行以下の計算をサブルーチンにしたのは、あとのプログラムでも、この計算が使われるということを予想していたからです。そういうことを考えなければ、何もここでサブルーチンを使わずに、90行と100行の間に1000行～1090行まで（1100行はぬかすことが必要）を入れてもよいのです。

注意すべきこと

このプログラムについてもう少し詳しくみていくことにしましょう。

このプログラムを読むときに注意すべきことは、1010行のIF文が、どのような働きをしているかです。この文によってM（何月）が3以上ならば1050行に飛びます。しかし、IF文のきまりからMが3以上でなければ（3未満なら）次の文に進みます。このようにしてMが3以上のとき（3月から12月）には③の部分を、Mが3未満のとき（1月と2月）には④の部分を行わせることができます。

ここで、1040行のGOTO文が重要な働きをしているということに注意を払っておいてください。もし、1040行のGOTO文がなかったらどうなるでしょうか。

たとえば、1985年1月1日の曜日を求めたいと思ったときには、Mは1ですから④の部分が実行されます。しかし、1040行を忘れていると、④を実行したすぐあとに1050・1060行の③の部分も実行されてしまうのです。そこで、M1は、1020行で11という正しい値になりますが、そのあとの1050行ではあらたに−1という数をあたえられ、その値で次の計算がなされてしまうことになるわけです。

また、フローチャートでは1行に書かれていた⑤の式が、このプログラムでは、1070と1080行の2つの文に分けられていることにも注意してください。ベーシックは1行に1つの文を書くことが普通で、何行にも続けて文を書くことはできません。そこで、長い文で1行に書ききれないようなときには、このように2つの文、あるいは、それ以上に分けて書くのです。

1070行で最初の3つの項を一度Jに代入しておきます。そして次の1080行で、そのJの値と残りの3項の値を加えて、その和を新しいJの値とするのです。これで結局フローチャートの⑤の式と同じ計算をしたことになります。1080行では、左辺と右辺に同じJがありますが、これは、数学で使われる等式ではなく、代入を意味している式であるということを思い出してください。

また、1090行のような式もよく使われます。これは、Jを7で割ったあまりを求める式です。INT(J/7)でJを7で割った商が出ますから、それを7倍してJから引けばあまりが出ます。そのあまりは、当然0から6の整数です。このあまりがIの値になり、それが曜日を示すのです。このIの値は、あとに結果を出力するとき(170行)にも使われることを覚えておいてください。

110行から170行までは、結果を出力する部分です。120～150行では、FOR—NEXT文とREAD—DATA文によって、文字配列B$(6)のB$(0)からB$(6)までの、それぞれにSUN(日曜)からSAT(土曜)までの3文字づつを入れています。そして、170行のPRINT文では、この文字配列B$とサブルーチンで求めたIの値を使って結果を印刷します。

このプログラムでは、文字配列変数を使いましたが、7つの曜日のそれぞれをPRINT文によって出力させてもよいのです。そうすれば、B$(6)のような配列は不要になりますから、20行のDIMも不要になります。そのようにして作ったプログラムは図4—27に示しておきました。図4—27のプログラムを図4—26(a)の110行以下と入れかえても、まったく同じ結果が得ら

```
110 REM
120 ON I+1 GO TO 130,150,170,190,210,230,250
130 LET C$="SUN"
140 GOTO 260
150 LET C$="MON"
160 GOTO 260
170 LET C$="TUE"
180 GOTO 260
190 LET C$="WED"
200 GOTO 260
210 LET C$="THU"
220 GOTO 260
230 LET C$="FRI"
240 GOTO 260
250 LET C$="SAT"
260 REM
270 PRINT Y; "NEN" ;M; "GATSU" ;D; "NICHI"
280 PRINT TAB(10); "* *  ";C$;"* *"
290 STOP
```

図4－27　曜日を調べるプログラムの一部

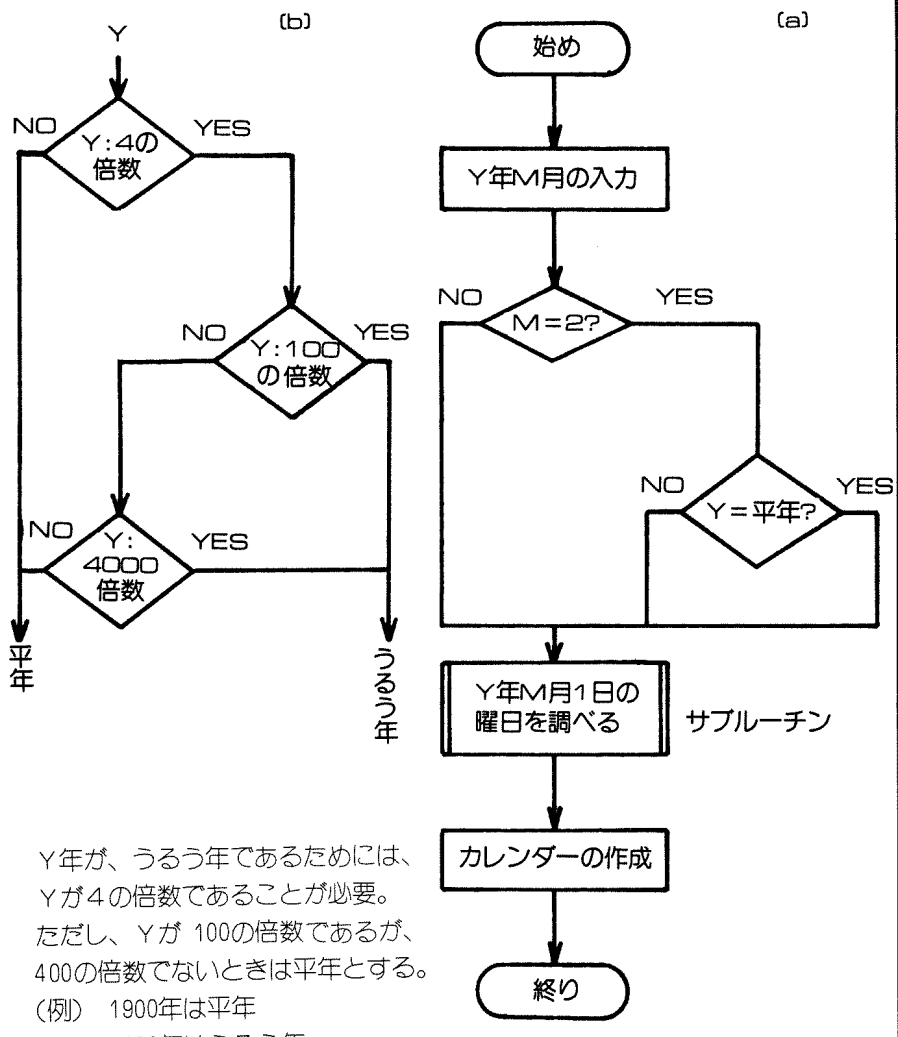
れます。ここでは、120行にON・GOTO文を使っています。ON・GOTO文に使われる変数値は、4捨5入して1以上の整数であるので、I＋1として計算しています。

ある月のカレンダー

曜日を調べるプログラムは以上のように完成しましたが、これだけでは、やはりおもしろくありません。次には、ある月のカレンダーを画面に出力させるようなプログラムを作ることにしましょう。

ある月のカレンダーを描くためには、何を知っていなければならないでしょうか。手近にあるカレンダーをながめてみるとわかると思いますが、カレンダーを描くためには、次の2つのことがわかっていなければなりません。

(1)　その月の1日は何曜日か。



Y年が、うるう年であるためには、
Yが4の倍数であることが必要。
ただし、Yが 100の倍数であるが、
400の倍数でないときは平年とする。
(例) 1900年は平年
2000年はうるう年

図4－28 うるう年と平年

(2)　その月は何日まであるのか。

ここで、(1)については、前に作った曜日を調べるプログラムを使うことが可能なようです。また(2)については、大の月(31日まである)と小の月(30日まである)は決まっていますから、それをおぼえておけばよいわけです。ただし、2月は、その年がうるう年であれば29日まで、平年であれば28日までですから、それも調べる必要があります。

カレンダー作成のためのフローチャートは、図4—28(a)の通りです。Y年M月1日の曜日を調べる部分は、前のプログラムをサブルーチンとして使うことにしましょう。したがって、カレンダーを実際に描く部分は今のところ除外して考えると、このプログラムですこし難しいのは、それぞれの月が何日まであるかを調べるところだけです。とはいっても、年によって日の数が違うのは2月だけですから、その年がうるう年か平年かがわかればいいことになります。

うるう年の求め方

うるう年を求めるのが面倒なのは、図4—28(b)に示したような例外があるためです。うるう年か平年かを調べるフローチャートをくわしく書くと(b)図のようになります。このように判断分岐がかさなるとわかりにくくなりますが、頭を整理しながら、プログラミングしてみると、212頁の図4—29、213頁の30、31のようになります。このプログラムは、主として次の3つの部分に分かれています。

(1)　年・月の入力とカレンダーの出力をするメインプログラム(メインプログラムとは、サブルーチンに対していわれる名称で、サブルーチンを呼び出す側の方をさす)。ここで、900～920行に小さなサブルーチンがあるが、これはカレ

```
 10 REM
 20 DIM E(12)
 30 PRINT "YEAR ?"
 40 INPUT Y
 50 PRINT "MONTH ?"
 60 INPUT M
 70 REM
 80 GOSUB 2000→サブルーチンB
 90 REM
100 LET D=1
110 GOSUB 1000→サブルーチンA
120 REM
130 PRINT "  * * " ;Y; "NEN" ;M; "GATSU"
140 GOSUB 900
150 PRINT TAB(4);" SUN MON TUE WED THU FRI SAT"
160 GOSUB 900
170 FOR K=1 TO 6
180 FOR L=0 TO 6
190 LET S=7*(K-1)+L
200 IF S<I THEN 270
210 LET S1=S-I+1
220 IF S1>E(M) THEN 300
230 LET T=1
240 IF S1<=9 THEN 260
250 LET T=0
260 PRINT TAB(4*L+T+5);S1;
270 NEXT L
280 PRINT
290 NEXT K
300 REM
310 GOSUB 900
320 STOP

900 REM
910 PRINT TAB(4); "----------------------"
920 RETURN
```

入力

L

K

数字の出力

図4－29　メインプログラム

```
1000 REM
1010 IF M>=3 THEN 1050
1020 LET M1=M+10
1030 LET Y1=Y-1
1040 GOTO 1070
1050 LET M1=M-2
1060 LET Y1=Y
1070 LET J=Y1+INT(Y1/4)-INT(Y1/100)
1080 LET J=J+INT(Y1/400)+INT(2.6*M1-0.2)+D
1090 LET I=J-7*INT(J/7)
1100 RETURN
```

図4－30　サブルーチンA(曜日を求める)

```
2000 REM
2010 FOR K=1 TO 12
2020 READ E(K)
2030 NEXT K
2040 DATA 31,28,31,30,31,30,31,31,30,31,30,31
2050 REM
2060 IF M<>2 THEN 2110
2070 IF Y/4<>INT(Y/4) THEN 2110
2080 IF Y/100<>INT(Y/100) THEN 2100
2090 IF Y/400<>INT(Y/400) THEN 2110
2100 LET E(2)=29
2110 RETURN
```

図4－31　サブルーチンB(各月の日数を求める)

ンダーの枠を出力するために使うもので、さほど重要ではないのでメインプログラムに含めて考えてよい。

(2)　曜日を決めるサブルーチンA。このプログラムは、図4－26のサブルーチンをそのままここに持ってきたものである。

(3)　各月の日数を決めるサブルーチンB。ここでは、配列Eに、平年の各月の日数を入れたあと、その求めたい年がうるう年かどうかと図4－28(b)のフローチャートにそって調べ、うるう年であれば2月の日数（E

(2)に入れられている）を29日にするという仕事をする。（注。20行で、DIM E(12)という宣言をしているから配列としてはE(0)からE(12)まで13個のものが使える。したがって本来なら、DIM E(11)で十分なのだが、プログラムをわかりやすくするために、E(0)は使わなかった。）

この3つの部分は、(1)の入力、(3)のサブルーチンB、(2)のサブルーチンA、(1)の結果の出力という順序で実行されることになります。

前述のように、月の始め（1日）の曜日を求めるためのサブルーチンAとしては前に作ったプログラムを、そのまま利用しています。しかし今回のプログラムで入力するのは、年(Y)・月(M)だけで日(D)は入力しないため、ここでは、110行でサブルーチンAを呼ぶ前に100行でDに1という値をあたえています。これによって、Y年M月1日の曜日を求めることができるのです。

最後に、130～310行が実行され、カレンダーが出力されます。ここでは少し複雑なことを行っているので、くわしく説明をします。

カレンダーの出力形式

130～160行までと310行はカレンダーの枠や年月曜日などを出力するためのPRINT文です。ここで横線は同じものを3本引くことになるので900行からのサブルーチンにしました。

このようにサブルーチンは、同じことを何度もするときそれをまとめてつくるときと、サブルーチンA、Bのように、1度しか実行されないけれど、プログラムとしてひとつのまとまりの形をしているのでつくるときのふた通りがあります。

次には、1日から月末までの数字を出力します。このとき、横は日曜から土曜までの7日分、縦は最も長く必要になるときを考えて6行分取

(a)

	NEN		GATSU				
	SUN	MON	TUE	WED	THU	FRI	SAT
K=1	(0)	(1)	(2)	(3)	(4)	(5)	(6)
2	(7)	(8)	(9)	(10)	(11)	(12)	(13)
3	(14)	(15)	(16)	(17)	(18)	(19)	(20)
4	(21)	(22)	(23)	(24)	(25)	(26)	(27)
5	(28)	(29)	(30)	(31)	(32)	(33)	(34)
6	(35)	(36)	(37)	(38)	(39)	(40)	(41)
	L=0	1	2	3	4	5	6

( )内はSの値　S=7×(K−1)+L

(b)

1900 NEN　2 GATSU

SUN	MON	TUE	WED	THU	FRI	SAT
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28			

(c)

```
PRINT TAB(5);A
A=4のとき　・・・・・・4
A=14のとき・・・・・・14
          |   |↑
          5   符号用空白
```

図4−32　カレンダーの出力形式

っておきます。この縦と横の関係をとりあっかっているのが、170〜290行と180〜270行のFOR・NEXT文です。この場合、まず内側のLが0から6まで1つづつ変わり、次に外側のKが変わることになります。

この2つのFOR−NEXT文のループによって、図4−32(a)にあるような、6×7の42コの空白のそれぞれに100行で定義された値Sをつけることができます。この値Sをうまく利用することによってカレンダーを描くことができるのです。

次は(a)図の(0)から

(6)までのどこから始めて、その月の日数分だけ数を置いていけばカレンダーが完成します。どこから1日を始めるかは、1000行からのサブルーチンAで求めたIの値を使います。

このプログラムを使って、1900年2月のカレンダーを出力してみました。図4—32(b)がその結果です。このように、きれいに数を並べるために260行でTABを使っていますが、図4—32(c)に示したように、TABは数字をその頭の部分でそろえるので、1けたの数（1日から9日まで）と2けたの数（10日以降）が、きれいに並ぶようにするためにはちょっとした技術が必要です。そのために使われているのが、230～260行の変数Tです。

変数Tには230行で1の値があたえられます。TはS1（これが日を表わす）が9以下のときには1のまま、9より大きいときには0になります。

したがって、260行のTABにおいては、日が1けたのときには、1つ余分にTABが進められて2けたのときと数がそろうようになるのです。

今日を中心としたカレンダー

カレンダーは非常に便利なものですが、これだけではふつうの紙に印刷されているカレンダーと何ら変るところがありません。マイコンの持ついろいろな能力のうちの1つ、数を並べたりくっつけたりする力を使って、「今日」を中心としたカレンダーを作ってみることにします。

カレンダーでは1月が単位となっていて、1日から月末まで全く同じ画面が写っていました。

しかし私たちはいつも1月を単位にして生きているのではありません。「今日から2週間後」

とか、「あの日から10日前」のように、ある日を中心にして考えたりすることも多いのです。こんなとき、1月が単位のカレンダーでは、前のページをめくったり、次のページをめくったりしなければならず、なかなか面倒くさいものです。

そこで、今日が月の始めであっても、月末であっても、とにかく「今日」が画面の中央部にくるようなカレンダーがあるとしたら便利でしょう。そこで、そういうカレンダーを作ってみることにしました。

このカレンダーも基本は前述のものとあまり変りません。カレンダーで空白にしていた部分に、前の月あるいは次の月の日付を入れることを考えればよいのです。

このプログラムでも、前に作った曜日・日数を求めるサブルーチンA・Bを使っています。

このプログラムのフローチャートは218頁の図4—33、プログラムは220頁の図4—34に示してあります。

注意すべきこと

ここで注意すべきことは、日付を印刷する際に使われる変数C（2）のプログラムではS1）がどのようにふやされていくかです。図4—33(b)では、図4—34(b)のプログラム290～450行と370～430行の2重のFOR・NEXTループの中でサブルーチンとしてよばれてCの値をひとつづつふやしていく部分の詳細なフローチャートを掲げました。ここでコネクタaはサブルーチンの入口をしめし、コネクタb（2ヵ所あります）は出口をしめしています。

図4—33の(a)と(b)を比較すれば明らかですが、(b)と(a)の計算部分はちょうど逆の作業をしています。(a)においては、カレンダーの先頭に入る

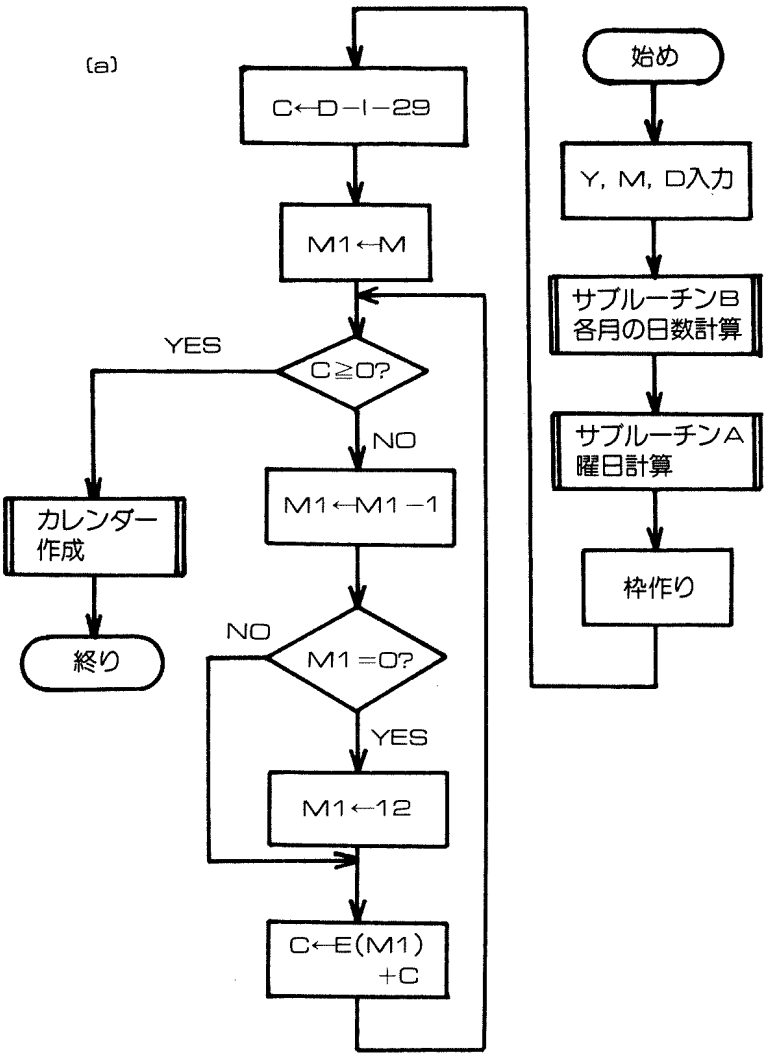


図４－33(ａ)　今日を中心としたカレンダーのフローチャート

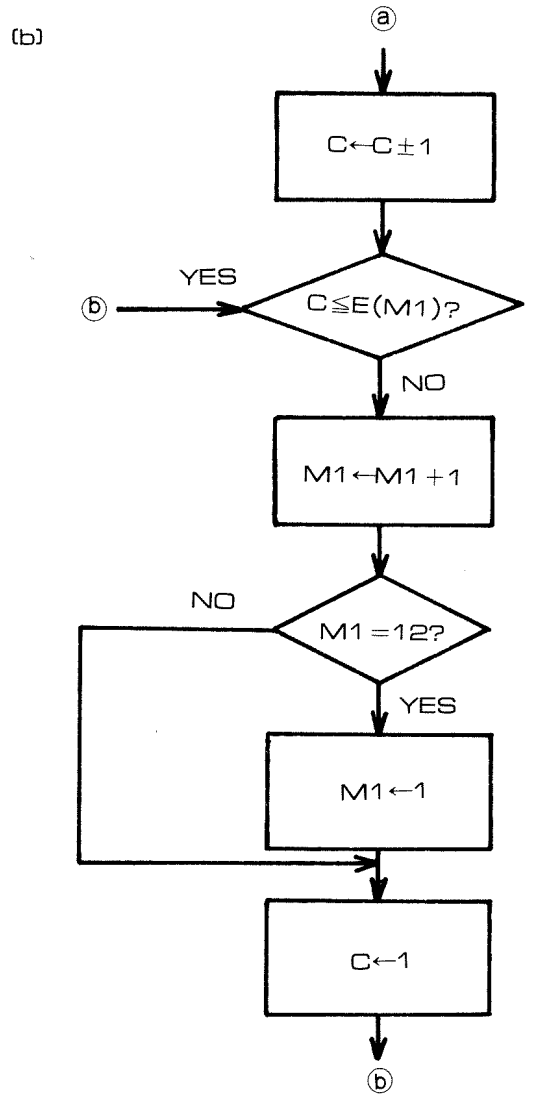


図4－33(b)　今日を中心としたカレンダーのフローチャート

図4－34　今日を中心としたカレンダーのプログラム

(a)
```
 10 REM
 20 DIM E(12)
 30 PRINT "YEAR ?"
 40 INPUT Y
 50 PRINT "MONTH ?"
 60 INPUT M
 70 PRINT "DAY ?"
 80 INPUT D
 90 REM
100 GOSUB 2000
110 REM
120 GOSUB 1000
130 REM
140 PRINT "  * *";Y; "NEN" ;M; "GATSU" ;D; "NICHI  * *"
150 GOSUB 900
160 PRINT TAB(4);" SUN MON TUE WED THU FRI SAT"
170 GOSUB 900
180 REM
190 LET C=D-1-29
200 LET M1=M
210 IF C>=0 THEN 280
220 LET M1=M1-1
230 IF M1<>0 THEN 250
240 LET M1=12
250 LET C=E(M1)+C
260 GOTO 210
```

(b)
```
800 REM
810 LET C=C+1
820 IF C<=E(M1) THEN 870
830 LET M1=M1+1
840 IF M1<=12 THEN 860
850 LET M1=1
860 LET C=1
870 RETURN
```

```
 900 REM
 910 PRINT TAB(4);" ----------"
 920 RETURN

 930 REM
 940 LET T=1
 950 IF M1<=9 THEN 970
 960 LET T=0
 970 PRINT "(";TAB(T+1);M1;")";
 980 RETURN

1000 REM
1100 RETURN
```
サブルーチンA(曜日を決める)
図4－30参照

```
2000 REM
2110 RETURN
```
サブルーチンB(各月の日数を決める)
図4－31参照

(c)
```
 280 REM
 290 FOR K=1 TO 9
 300 IF K<>1 THEN 320
 310 GOSUB 930
 320 IF K<>5 THEN 340
 330 PRINT "-->";
 340 IF K<>9 THEN 360
 350 GOSUB 930
 360 REM
 370 FOR L=0 TO 6
 380 GOSUB 800
 390 LET T=1
 400 IF C<=9 THEN 420
 410 LET T=0
 420 PRINT TAB(4*L+T+5);C;
 430 NEXT L
 440 PRINT
 450 NEXT K
 460 GOSUB 900
 470 STOP
```

Lのループ

Kのループ

＊＊ 1901 NEN 5 GATSU 15 NICHI ＊＊

	SUN	MON	TUE	WED	THU	FRI	SAT
(4)	14	15	16	17	18	19	20
	21	22	23	24	25	26	27
	28	29	30	1	2	3	4
	5	6	7	8	9	10	11
→	12	13	14	15	16	17	18
	19	20	21	22	23	24	25
	26	27	28	29	30	31	1
	2	3	4	5	6	7	8
(6)	9	10	11	12	13	14	15

図4－35　今日を中心としたカレンダー

べき数字を探すためにCの値をどんどんさかのぼらせていっています。そしてみつかったCの値から始まって、1つづつCの値がふやされ、それが日付として出力されるのです。そのとき、Cの値がある月の日数を越えると次の月の1日に戻り再び1つづつふえていきます。

このようにして得られたカレンダーが、図4—35に示されています。このカレンダーは1901年5月15日を中心として、その前後、ほぼ4週間を出力したものです。左端の括弧内の数字は、その行の先頭の日が属している月です。したがって、図4—35の例では、4月14日から6月15日までのカレンダーが同時に出力されているわけです。

〈著者紹介〉

谷本敏一（たにもととしかず）
一九五〇年　北海道に生まれる
東京大学文学部心理学科卒業
心理学・教育学におけるコンピュータの応用を研究

野島久雄（のじまひさお）
一九五六年　埼玉県に生まれる
東京大学大学院修士課程
コンピュータのソフトウェア理論と応用を研究

マイコンの使い方遊び方

1981年11月25日　発行　©

著者　谷本敏一
　　野島久雄
発行者　富永弘一
印刷所　慶昌堂印刷株式会社

発行所　東京都台東区台東2丁目24　株式会社新星出版社
電話（831）0743　郵便番号110　振替東京 4-72233

趣味の技術入門シリーズ

- やさしい **電子オモチャの作り方** ——明翫泰平著 定価680円
- よくわかる **マイコン入門** ——山科敦之・吉田紀彦共著 定価680円
- よくわかる **マイコン使い方・遊び方** ——谷本敏一・野島久雄共著 定価780円
- **初歩のマイコン入門** ——三宅　誠・佐藤清明共著 定価780円
- よくわかる **BCL入門** ——大河原　真著 定価680円
- よくわかる **アマチュア無線入門** ——大塚正幸著 定価680円
- よくわかる **シンセサイザー入門** ——平塚　肇著 定価780円
- よくわかる **オーディオ入門** ——森　耕一著 定価680円
- やさしい **ラジオ工作入門** ——黒岩文雄著 定価680円
- やさしい **ラジコン飛行機入門** ——畠山　泉著 定価680円
- やさしい **ラジコン自動車入門** ——水谷則雄著 定価680円
- よくわかる **鉄道模型入門** ——峰岸則行著 定価680円
- よくわかる **天体観測入門** ——渋谷恵一著 定価780円

●よくわかる　ＢＣＬ入門　★大河原真著
●よくわかる　オーディオ入門　★森　耕一著
●よくわかる　マイコン入門　★山科敦之著
●やさしい　ラジオ工作入門　★黒岩文雄著

趣味の技術入門

●よくわかる マイコンの使い方遊び方

新星出版社
定価780円
2055 35211 3151